Litton
TASC

AICE Contractor Report 9569-01
CLIN 0005/A0001

Scalable Techniques for Large Scale Dynamic Channel Building

Final Report for the period 1 March 1999 to 31 March 2000

Brian DeCleene

Litton/TASC
55 Walkers Brook Dr.
Reading, MA 01867

Prepared for
DARPA/ITO under Contract No. DABT63-99-C-0018

April 19, 2000

ABSTRACT

This report summarizes the research and software artifacts produced as part of the Agile Information Control Environment (AICE) program in the area of real-time content-based information dissemination through resource-constrained channelization. By identifying opportunities to multicast shared information needs to multiple users simultaneously, channelization increases the information throughput of information dissemination system while balancing the sender/receiver processing load.

Multiple channelization algorithms are presented within this report and their characteristics explored. A mathematical framework for analyzing these algorithms is outlined and corresponding measures of effectiveness defined. Besides theoretical analysis, the report outlines simulation-based models and operational tools used to quantify the various algorithms' performance within the context of the theoretical measurements. From our analysis and simulations based on operational missions, the report concludes that channelization can reduce the network load while increasing the information throughput by more than 100% compared to traditional unicast stove-pipe approaches.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. SUMMARY

The mission of the DARPA/ITO Agile Information Control Environment (AICE) program is to research and quantify techniques for maximizing the utility of mission-critical information delivered over finite heterogeneous network and end-host resources. The utility of information is defined as the value of the information to the user; not the bandwidth or other resources consumed by transmitting the information. The program also provides a command and control capability that allows commanders to define policies dictating how resources should be allocated when conflicts occur.

Under AICE, the Scalable Techniques for Large Scale Dynamic Channel Building (ChannelTech) project conducted by TASC investigated and characterized approaches for constructing shared multicast channels that improved information throughput. From this investigation, TASC developed a mathematical framework for characterizing channelization algorithms; simulation models for characterizing performance; and operational software that is being incorporated into the near-term fielded Information Dissemination Management (IDM) system as part of the GCCS.

The analysis defines the channelization problem as an optimization problem over a set of sets with individual "utility" measures on each set. Nominally, the resulting channels satisfy properties such as orthogonality and precision that are summarized in this report and described in full within the published papers in the appendix. Through the software simulation tools (also described in this report), channelization algorithms are shown to improve information throughput. However, the performance of different channelization approaches varies with the system architecture and user constraints.

2. INTRODUCTION

The Scalable Techniques for Large Scale Dynamic Channel Building (ChannelTech) project is a three-year research and development effort conducted by TASC under the DARPA/ITO Agile Information Control Environment (AICE) program. The objective of the ChannelTech project is to develop fast, computationally tractable algorithms for the construction of channels that maximize the utility of information delivered to the user subject to resource limitations.

This work is currently organized into a base task and four options that provide increasing performance and capabilities. Of these tasks, this report summarizes the work performed on the funded base and first optional tasks. They are:

- Task 1 - Base: Provide support for Functional Architecture Control Board (FACB), System Engineering Control Board (SECB), and Performance, Analysis & Integration (PA&I) meetings.
- Task 2.1 - Initial Static Channelization Techniques: Six-month activity to design and demonstrate analysis infrastructure and incorporate it within the AICE functional architecture. Develop performance models for basic channelization approaches under static loading conditions. Analyze and characterize the performance characteristics of the channelization algorithms.

The out-year research activities listed below were not executed due to the early termination of the AICE program by the government.

- Optional Task 2.2 – Advanced Static Channelization Techniques: Design and demonstrate advanced channelization approaches under static loading. Incorporate constrained optimization algorithms including source density models and network resource limitations. Investigate computational representations of information requests to enable abstract semantic processing. Analyze and characterize the performance characteristics of these algorithms.
- Optional Task 3 – Dynamic Channelization Techniques: Design and demonstrate channelization approaches under dynamic loading conditions. Analyze and characterize the performance characteristics with emphasis on real-time scalable processing.
- Optional Task 4 – Multiresolution Channelization Techniques: Design and demonstrate highly-scalable algorithms using multiresolution approaches to reduce problem complexity.

This final report summarizes the approach and accomplishments of the ChannelTech project. The remainder of this section provides the executive summary of the project structure and results. Specifically, Section 1.1 defines the relationship of this work within the context of the AICE functional architecture. Then, Section 1.2 describes TASC's structured approach to research and development. Finally, Sections 1.3 and 1.4 summarize the major accomplishments and project outputs.

In Section 2, the problem is formulated and the solution algorithms are defined along with metrics for quantifying the performance. Section 3 provides details about the experimental and operational software developed under ChannelTech. Section 4 outlines the Experimentation and Comparative Analysis that was accomplished. Sections 5 through 7 contain the project summary, bibliography, and acronym list, respectively.

Appendix A contains manuals for software tools used during the research. Appendix B contains papers and articles related to the topic.

2.1 RELATIONSHIP WITHIN AICE FUNCTIONAL ARCHITECTURE

The mission of the AICE program is to research and quantify techniques for maximizing the utility of mission-critical information delivered over finite heterogeneous network and end-host resources. The utility of information is defined as the value of the information to the user; not the bandwidth or other resources consumed by transmitting the information. The program also provides a command and control capability that allows commanders to define policies dictating how resources should be allocated when conflicts occur.

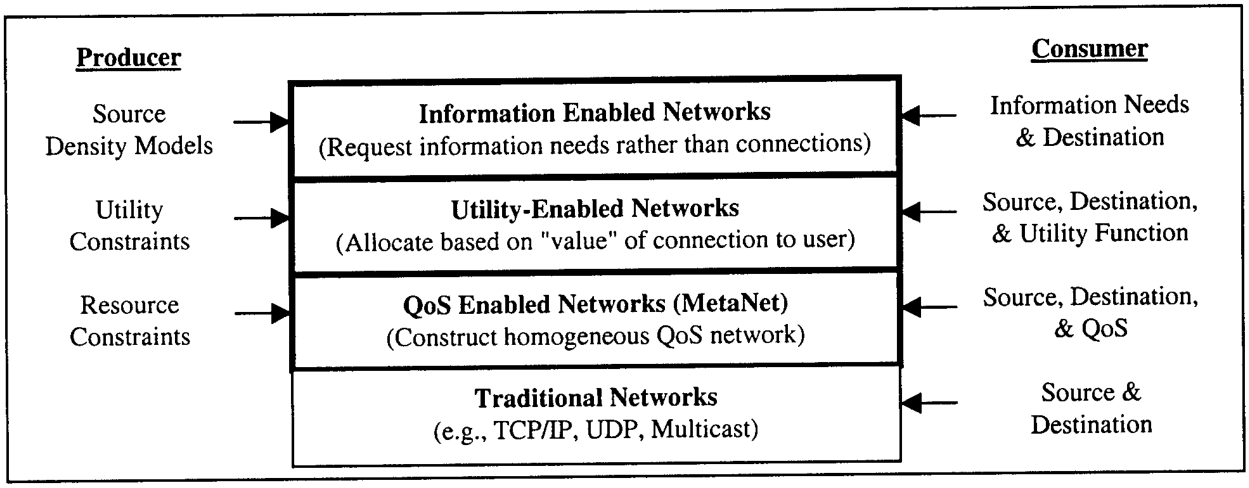

Producer
Source Density Models
Utility Constraints
Resource Constraints
Information Enabled Networks
(Request information needs rather than connections)
Utility-Enabled Networks
(Allocate based on "value" of connection to user)
QoS Enabled Networks (MetaNet)
(Construct homogeneous QoS network)
Traditional Networks
(e.g., TCP/IP, UDP, Multicast)
Consumer
Information Needs & Destination
Source, Destination, & Utility Function
Source, Destination, & QoS
Source & Destination

Figure 1-1: Functional Layers of the AICE Program

Addressing the issues surrounding AICE, the program is roughly organized into three functional layers shown in Figure 1-1. They are:

- MetaNet Services: Addresses the homogeneity of networks by constructing a virtual network with resource reservation capabilities.
- Utility-Enabled Networking (AIC[1]): Allocates network connections based on the "value" or "utility" of a particular set of QoS parameters to the user. This allows the system to make responsible decisions on when and how to degrade a particular connection to maximize the overall success of the missions.

[1] The acronyms for the components (i.e., AIC, AEI, and IPM) are based on historical discussions within the functional architecture and no longer accurately reflect the functionality performed at the specified layer. While we retain this notation for consistency across documentation, we have modified their names for the sake of clarity.

- Information-Enabled Networking (AEI): Performs network optimization based on information about the requested content as well as the utility. This allows resources to be shared across connections and removes the requirement that the user must understand the network requirements for the desired connection.

In addition, policy services (IPM) ensure that service requests comply with resource allocation doctrine defined by commanders. These services may exist at the AIC layer where utility-based connections are scaled according to mission importance as well as at the AEI layer where policies manage information access and flow.

At the AEI layer, the availability of metadata about the desired content combined with external system information allow for the construction of channels that multicast data to relevant users while maximizing the achieved utility. *The ChannelTech R&D program investigates channelizations that achieve very fast, near-optimal solutions for the quiescent (static) and extremely fast quasi-optimal solutions for the incremental (dynamic) cases and have computational complexity O(N log N) for N users.*

2.2 SUMMARY OF ACCOMPLISHMENTS

A white paper titled *Utility-Based Management of Information Dissemination* [2] was completed and is included in Appendix B. This paper proposes a modified mathematical definition to the functional architecture and formulates the channelization problem as a resource optimization problem. Included in the paper are definitions of key channelization properties such as orthogonality, exactness, and precision along with proofs for algorithm characteristics for certain classes of channel algorithms.

TASC has also completed initial analysis, simulation, and experimentation of the contaiment algorithm (see Section 4.5.3). Simulation results of the channelization algorithms reveal that, for large numbers of users, a set of orthogonally disjoint channels are constructed. This work is captured in a white paper titled *Examining the Containment Algorithm* [3]. Validating this analysis, an experiment with 1000 information requests was performed and resulted in approximately 60 channels within the network versus 1000 channels in the absence of channelization.

TASC has released and successfully installed, unit tested, and demonstrated the first release of the ChannelTech software at the PAC99 laboratory. Software highlights include:

- Utility-based information requests and War-Fighter Editor: Leveraging prior work performed under DARPA/ISO BADD II, support of utility-based information requests was developed. With the modifications, users can construct information requests based on a dynamically loaded schema and command hierarchy, specify their information interests and connection characteristics, and submit them for channelization. Preliminary hooks to policy and other AICE layers are provided.
- Dual Channelization Algorithms: Two channelization algorithms were developed during this reporting period. Under the trivial algorithm, each information request is assigned to a single channel. On the other hand, the containment algorithm aggregates any information request that is fully satisfied by another request onto a

shared channel. These algorithms have been tested to support in excess of 3000 information requests.[2]

- Experimental Test Driver and Examples: A test harness that allows users to construct sample information request scenarios was developed. For example, one script submits random requests for information at a fixed rate and then sequentially removes them. These provided the basis for the PAC99 experimentation.
- Log Analysis Tool: This tool performs post-analysis of the log files to quantify the performance characteristics of the channelization algorithm. Information extracted includes channelization gain (the reduction of number of network requests due to channelization); channel orthogonality; and channel precision. The tool also measures our performance characteristics of the software such as memory consumed and end-to-end delay.

In refining the functional architecture, TASC released a Unified Modeling Language (UML) model of the channelization components. This model contains detailed information about the representation of information requests or profiles to the channelization services and provides visibility into the operations performed by these services. This model also provides a cross-program integration plan between the DARPA/ISO BADD program that addressed basic information dissemination services and the DARPA/ITO AICE program that focuses on highly scalable information-enabled resource allocation algorithms. The latest version of this model can be found at the AICE website (http://aice.trw.com).

2.3 SPECIFIC OUTPUTS AND TECHNOLOGY TRANSFER

Table 1-1 lists the specific technical products generated by the ChannelTech project. The details of these outputs are discussed in the section referenced in the last column. In addition, numerous technical briefings have been provided throughout the project and are available upon request.

Table 1-1 - Technical Products Generated by ChannelTech

Output	Type	Description	Section/ Appendix
Utility-Based Management of Information Dissemination	Report	Mathematically formalizes utility-based information and proves key properties.	B
Examining the Containment Algorithm	Report	Quantifies the characteristics and performance of the coverage algorithm.	B
Real-time Information Management Environment (RIME)	Abstract	Description of the software architecture and performance characteristics. Submitted to SPIE AeroSense.	B

[2] For comparison, the BADD system (ver. 4.3.1) which provides push-based delivery but lacks sophisticated channelization or resource management can only support between 300 and 500 requests before suffering serious performance degradation.

Output	Type	Description	Section/ Appendix
AEI Functional Architecture UML Model	Model	UML model describing the functional components of the RIME system. Developed in conjunction with the Arch. Control Board.	http://aice.trw.com
RIME Simulation Tool	Software	Tool for prototyping channelization algorithms and visually inspecting characteristics.	A.1
CPcrib Tool	Software	A Monte-Carlo simulation for exploring the theoretical properties of the various algorithms.	3.2.1/B
RIME Editor v1.0	Software	User interface enabling users on Unix and NT workstations to construct utility-based information requests. Support for hierarchical policy construction is also provided.	A.3
RIME Server v1.0	Software	Content-based resource management channelization algorithms using utility information. Also incorporates rudimentary policy services for managing resource allocation.	A.4
RIME Log Analysis Tool and Test harness	Software	Includes tools for driving the simulations and performing post analysis. Tool also allows for visual exploration of channelization characteristics at particular instances in time.	A.2

Technology transfer of the software is being accomplished, in part, through the DARPA/ISO BADD Phase II Transition program into the AITS/JPO IDM program. Specifically, ChannelTech improvements in request channelization have been integrated into version 4.6 of the BADD Core. This provides an order-of-magnitude performance improvement over the previous capability. Furthermore, AITS/JPO users may optionally use the testing harness software and analysis tools to better quantify system performance.

Research products such as reports and models have been conveyed to the research community through public postings and conferences. For example, a paper titled "Real-time Information Management Environment (RIME)" has been accepted to appear at SPIE AeroSense 2000 in April. This paper includes, in part, results from the ChannelTech analysis and experimentation.

3. METHODS, ASSUMPTIONS, AND PROCEDURES

The ChannelTech process for producing the technical outputs of the project consists of two sub-processes:

- Research Process: Represents two-thirds of the technical effort
- Software Development Process: Represents one-third of the technical effort

These two processes are coupled through a set of shared metrics that were used to compare the theory results against the actual capabilities implemented and transitioned. The overall organization of this final report coincides with this structure.

3.1 RESEARCH APPROACH

ChannelTech work was accomplished in two six-month research cycles. During each cycle the ChannelTech team is focused on exploring one or more well-defined research objectives through concept development, analysis, simulation, experimentation, and reporting. Certain ChannelTech concepts were also demonstrated at customer-sponsored meetings.

The R&D Process that ChannelTech follows is composed of several lower-level processes that are organized on two levels – program-level processes and research objective-level processes as described in the following:

Program-Level Processes

- Plan Program – This includes all activities required to define the scope of the overall program, to establish program milestones, to obtain resources, and to define roles and responsibilities. This is performed at the beginning of the program and is reviewed periodically.
- Develop Research Objectives – This is a technically oriented extension of the Program Planning process that is focused on defining the specific research objectives that need to be achieved in order for the Research Program to be considered successful. This includes top-level investigations to support definition of specific research objectives. This also requires periodic refinement of the Program Plan with respect to milestones and resources. This process will be conducted approximately every six months in order to define the direction of the project for the next research cycle.
- Report Findings – Periodically, at the conclusion of work during a research cycle, or at the conclusion of the entire research program, this process includes all work needed to document and present the work performed and the findings obtained. This document reports the research findings at the conclusion of the program.

Research Objective-Level Processes

- Investigate – This process includes all work required to locate, review, assess, assimilate, and apply the potentially related work of others to the research objective under study.

- Assess Current State of Knowledge – This process includes all activities performed to relate the prior work of others to the current research objective for the purpose of establishing a knowledge baseline.
- Develop Research Proposal – In this process the team:
  - Defines one or more hypotheses supporting a research objective,
  - Details the approach for proving or refuting the hypotheses,
  - Performs simulation to test the hypotheses,
  - Defines the specific methods and experiments needed to prove the hypotheses,
  - Identifies the resources and test environments required, and
  - States the results that are expected.

  For ChannelTech, the hypotheses are usually expressed as a set of algorithmic use cases.
- Conduct Experiments – This process includes all activities required to perform controlled tests to prove or refute a hypothesis.
- Prepare and Interpret Data – In this process the team transforms raw data from experiments into information upon which a subsequent analysis can be performed. This includes validating the data and removing or qualifying those entries judged to be invalid. It also includes interpreting and annotating the data with related information that may further qualify its validity and applicability.
- Conduct Analysis – This process involves the methodical examination of information to logically develop new knowledge related to the hypotheses under study.
- Develop Conclusions – This process includes all work required to relate newly developed knowledge to the hypotheses under study to determine whether the hypotheses were proven or refuted and hence, whether the research objectives were satisfied.

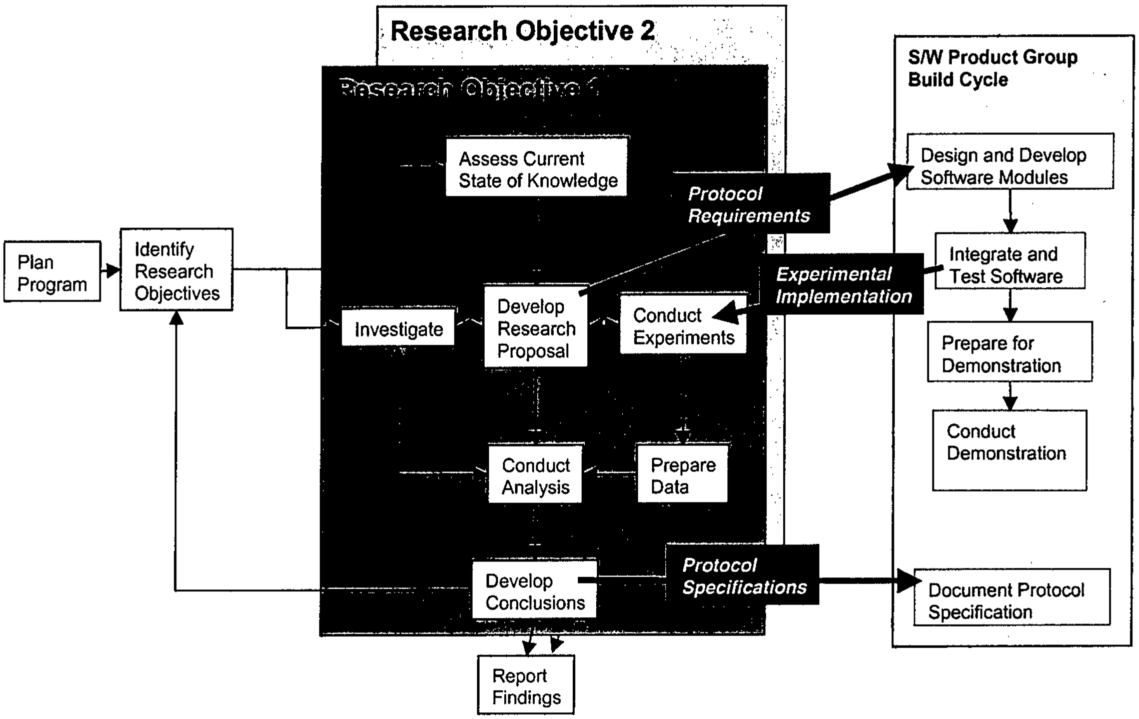

Research Objective 2
Research Objective 1
S/W Product Group Build Cycle
Plan Program
Identify Research Objectives
Assess Current State of Knowledge
Protocol Requirements
Design and Develop Software Modules
Experimental Implementation
Integrate and Test Software
Investigate
Develop Research Proposal
Conduct Experiments
Prepare for Demonstration
Conduct Demonstration
Conduct Analysis
Prepare Data
Develop Conclusions
Protocol Specifications
Document Protocol Specification
Report Findings

Figure 1-2: ChannelTech Project R&D Process Integrated with a Software Development Process

3.2 SOFTWARE DEVELOPMENT APPROACH

ChannelTech uses a tailored version of the TASC Software Process to support the development of software used for conducting experiments and for demonstrations. This software process (see Figure 1-2) is integrated into the R&D Process using the following build cycle:

- Design and Develop Software Modules
- Integrate and Test Software
- Prepare for Demonstration
- Conduct Demonstration.

Upon defining a research proposal, there is generally a need identified for the development of software that will demonstrate empirically the ideas supporting the research objective. Most of the software developed under ChannelTech instantiates a particular channelization algorithm. The protocol requirements for this software are usually expressed most effectively as pseudo-code or use cases demonstrating how the algorithm should operate. Agreement on these requirements by members of the ChannelTech team begins the software development process.

The software is developed to yield an experimental implementation that will be examined through exercising the ChannelTech prototype under controlled conditions. Often these controlled conditions also involve integration with external components that provide functionality outside the scope of ChannelTech. The software is verified by conducting several calibration trials to determine whether the results obtained agree with those developed during related analysis and/or simulation. The verified software implementation supports the Conduct Experiments process within the R&D Project Process.

Based upon the results of the experiments and the customer's schedule of meetings with Principal Investigators, the software may be enhanced with a graphical display or other mechanisms to provide a demonstration of the concepts developed toward meeting the research objective.

After conducting experiments, improving the protocol algorithms, and completing an analysis of the results, we draw conclusions and write the specification describing the algorithm. We will also deliver/distribute the software prototype outside of TASC, per contractual requirement.

4. PROBLEM FORMULATIONS AND DEFINITIONS

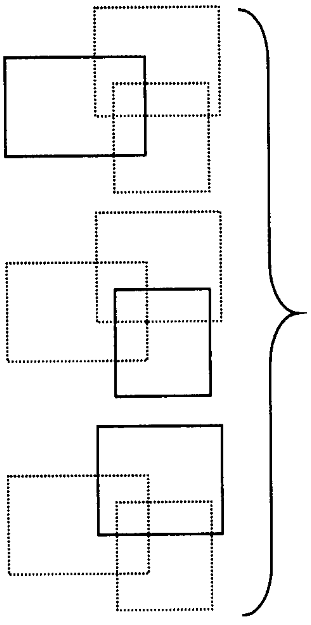
4.1 OVERVIEW

Achieving mission-critical situational awareness through information dominance requires the timely and efficient delivery of information in response to changing conditions. While programs such as BC2A and BADD have demonstrated the technical feasibility of information management and dissemination (IDM) and constructed an architectural framework, they have only begun to address the research issues necessary to achieve its full potential. The ChannelTech research investigates the techniques for improved resource optimization in the presence of constrained network and end-host resources.

For efficiency, delivery is performed using multicast whenever multiple users need the same data. Determining that multiple users need the same data (and hence that there is an opportunity to exploit multicast) can be accomplished by looking for overlaps in two or more user profiles. From a mathematical perspective, user profiles constitute sets in multi-dimensional space, and determining overlaps between user profiles is equivalent to computing set intersections: non-empty intersections provide multicast opportunities.

An obvious problem arises immediately in that for N sets (user profiles), there are 2^N unique subsets that can be constructed from intersections of a given user profile set (or its complement) with each of the other user sets. Hence even for a modest number of users (say, 150), an enormous number of subsets (2^{150}) must be evaluated to determine if, and to whom, the data should be multicast. The exponential relationship of the number of regions to the number of subregions is depicted in Figure 2-1 for a three-user case.

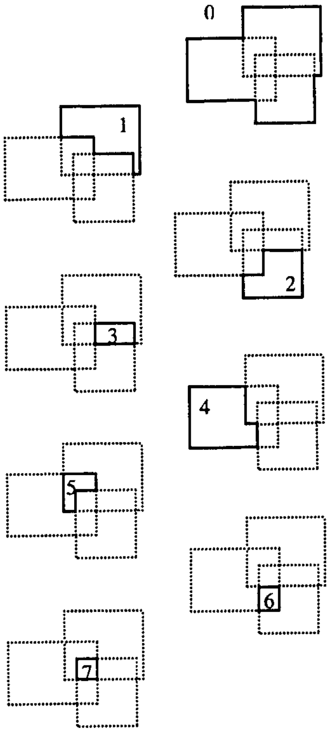
If only because explicit enumeration of a user base of this size exceeds the addressing range of current generation processors, direct approaches for assessing multicast possibilities are intractable. From a practical perspective, the problem is far worse: the number of possible multicast addresses that can realistically be supported by a network is more on the order of 2^{14} or less. Hence solutions must be developed for mapping the enormous number of potential multicast sets into a much smaller number that are in some sense optimal.



Informational needs for three users U_0, U_1, and U_2

I_2	I_1	I_0	Code	Operation
0	0	0	0	$I_2^c \cap I_1^c \cap I_o^c$
0	0	1	1	$I_2^c \cap I_1^c \cap I_o$
0	1	0	2	$I_2^c \cap I_1 \cap I_o^c$
0	1	1	3	$I_2^c \cap I_1 \cap I_o$
1	0	0	4	$I_2 \cap I_1^c \cap I_o^c$
1	0	1	5	$I_2 \cap I_1^c \cap I_o$
1	1	0	6	$I_2 \cap I_1 \cap I_o^c$
1	1	1	7	$I_2 \cap I_1 \cap I_o$

Decomposition and encoding of the collective informational needs

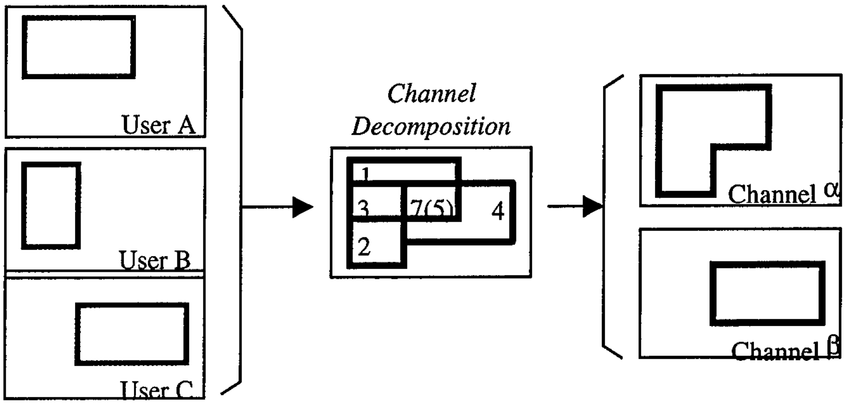


Relationship of encoded decomposition to user needs

Figure 2-1: Construction of the Set of User Information Need-Intersections and Corresponding Encoding for a Three-User Problem

4.2 MATRIX FORMULATION

The channelization problem can be recast mathematically in a number of different ways. For example, consider the following very simple, abstract model of an information dissemination system: There are N distinct information sets that are the a-priori decomposition of the M user requests into the disjoint subsets as described above. Consequently, each of the M users wants some collection of those sets. Let A be an $M \times N$ matrix, called the user request matrix, whose rows represent users and whose columns represent the decomposed information sets. If user i wants information set j, then the ij-th element of A is one, while if user i does not want information set j, then the ij-th element of A is zero. Analogously, we define the channelization matrix, C, as a 0-1 matrix with the ij-th element being one if and only if information set j is carried on channel i.



Channelization matrix (C) for the two channels in terms of the five regions is

$$\begin{array}{c}\alpha \\ \beta\end{array}\begin{bmatrix}1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0\end{bmatrix}^t$$

Subscription matrix (S) for users A,B, and C to the two channels is

$$\begin{array}{c}A \\ B \\ C\end{array}\begin{bmatrix}1 & 0 \\ 1 & 0 \\ 0 & 1\end{bmatrix}^t$$

Figure 2-2: Example Solutions for Matrix Formulation

When formulated in this fashion, the channelization problem is similar to a mixed-integer linear optimization problem. For example, if the subscription matrix, S, is another 0-1 matrix such that the *ij*-th element corresponds to the *i*-th user subscribing to the *j*-th channel, then we are interested in finding solutions to

$$CS \geq A \qquad (0.1)$$

that satisfy various constraints on the dimensions of C and other internal properties of C. An example is shown in Figure 2-2.

Because of the speed of simple matrix comparisons, this formulation has proven valuable in modeling many of the channelization algorithms; particularly as the number of requests becomes large. The software package **Cpcrib** is based upon this representation and is discussed in detail in *Experimental Software Description* [6].

4.3 STATE-SPACE FORMULATION

The channelization problem can also be formulated as a series of state changes based usually on the previous state alone. To see how this can be accomplished, we begin by observing that the 0-1 row of the user request or channelization matrices defined previously can be converted to an integer. For example, in the previous figure, channel α corresponds to an integer value of 23 (i.e., 10111) while channel β corresponds to a value of 24 (i.e., 11000). Thus, the current state of the system at any given time can be described by a list of integers that correspond to the various channel and/or profile definitions.

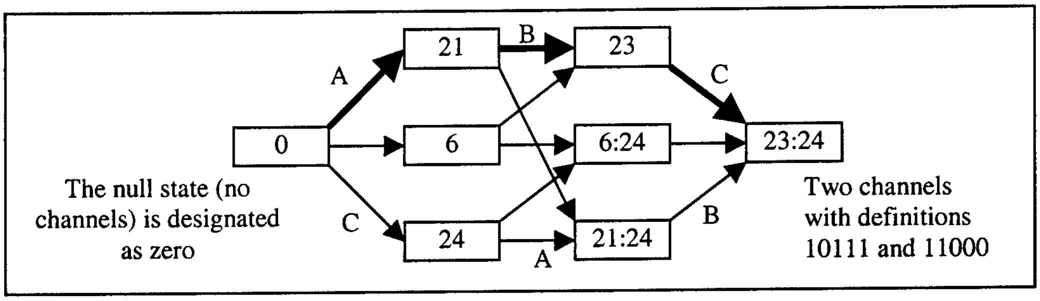

21
23
6
0
6:24
23:24
24
21:24
A
B
C
C
A
B
The null state (no channels) is designated as zero
Two channels with definitions 10111 and 11000

Figure 2-3: State Transitions for Channels Defined in Section 2.2

State transitions occur when a new information request is submitted. This is shown in Figure 2-3 for the channels defined in the previous section. For example, consider the bold line through the diagram. When user A submits his information request corresponding to regions 10101 with integer value 21, then a single channel is created with the same definition. The corresponding channel is shown as the state. Next, user B's request for regions 00110 (i.e., 6) generates an aggregated channel that is the union of their individual interests. The definition of this single channel is 23 and corresponds to channel α presented earlier. Finally when user C makes his final request for information, a second channel is constructed corresponding to channel β. Thus the final state is a pair of integers.

The advantage of this representation is that given the probability of being in a particular state at a given time along with the probability distribution for various transactions, the probability of being in each of the states can be calculated for all future submissions. From this, information such as the expected number of channels and standard deviation as a function of the number of information requests can be calculated. This approach is the basis for the RIMEStateSpace utility developed as part of ChannelTech and described in Section 3.2.2.

4.4 CHANNELIZATION PERFORMANCE METRICS

Quantifying the characteristics of various channelization algorithms is critical to performing a comparative analysis. In this section, we define key metrics in two categories: Channelization Algorithm and Software Performance Metrics. Specifically, in Section 2.4.1, Channelization Algorithm Metrics (CAM) describe and characterize the salient performance of a channelization algorithm without regard to how the algorithm was implemented. Then, Section 2.4.2 defines the Software Performance Metrics (SPM) and how they measure the effectiveness of a particular implementation of the algorithm.

4.4.1 Channelization Algorithm Metrics Definitions

In this section we define each Channelization Algorithm Metric as represented in the experiments and the RIME Log Analysis Tool.

Channel Count

Channel count is the number of channels, n_c, constructed given a certain set of information requests and is usually expressed as a function of number of requests, n_r. As we will see later,

- With the no aggregation algorithm, the number of channels always equals the number of requests ($n_c = n_r$).
- With the containment algorithm, $n_c \le n_r$.
- With the direct decomposition, $n_c \ge n_r$.

In this fashion, the channelization count measures the load imposed on the network in terms of the number of channels that need to be supported.

Experimentally, the number of channels depends upon, not only the number of requests, but also the definitions of those requests. Therefore, we will often consider the statistics about the count such as its expectation and deviation given a particular distribution of input requests.

Redundant Load

In addition to the network load due to the number of channels constructed, we need to measure the increase (or decrease) in the volume of data transmitted as a result of channelization. To measure this, we define the redundant load as the amount of data that is transmitted separately to multiple users.

We define the redundant gain as

$$G_r = \frac{\sum_{i=1}^{n_c} \sum_{j=i+1}^{n_c} \rho(E_i^c \cap E_j^c)}{\sum_{i=1}^{n_r} \sum_{j=i+1}^{n_r} \rho(E_i^r \cap E_j^r)}$$

where E^c and E^r are the exchange characteristics for the channel and request, respectively. The density function, $\rho(...)$, measures the load of a particular volume of data. Thus, the numerator of this equation measures the volume of duplicate data that is sent on separate channels while the denominator measures the volume of duplicate requests for data. If the channels are orthogonal (i.e., the channels do not overlap) then the gain is zero. However, a gain of zero does not imply that the channels are orthogonal — rather the measure of the overlaps may simply be zero.

Often it is difficult to define a-priori the density of information for the various regions of interest. For this reason, we define the approximate redundant gain as

$$\tilde{G}_r = \frac{\sum_{i=1}^{n_c} \sum_{j=i+1}^{n_c} 1(E_i^c \cap E_j^c)}{\sum_{i=1}^{n_r} \sum_{j=i+1}^{n_r} 1(E_i^r \cap E_j^r)}$$

where the indicator function, $1(...)$, replaces the density function and has a value of one if the two exchange characteristics intersect and zero otherwise. Unlike the previous equation, this is zero if and only if the set of channels is orthogonal.

Subscription Count

Whereas the channel count and redundant data measure the load imposed on the network, the subscription count measures the number of channel subscriptions necessary to satisfy a particular request. Let us define $S(i)$ to be the set of channel subscriptions to satisfy the i-th request. Then

$$n_s(i) = |S(i)|$$

where $|S(i)|$ is the number of elements in $S(i)$.

Excess Data

Analogous to the redundant gain that measures the amount of information that is transmitted redundantly, the excess gain measures the amount of data that a particular user receives that they did not request. Therefore, for the i-th user we define the excess gain as

$$G_e(i) = \frac{\sum_{E^c \in S(i)} \rho(E^c \cap \bar{E}_i^r)}{\rho(E_i^r)}.$$

The numerator of this equation measures the volume of data that was received but not requested whereby the denominator normalizes the result according to the total volume requested. We also define

$$\tilde{G}_e(i) = \sum_{E^c \in S(i)} 1(E^c \cap \bar{E}_i^r)$$

when the density of the information is not known a-priori. This second definition has the added property that it is zero if and only if the current channel set is exact for the specified user.

4.4.2 Software Performance Metrics Definitions

Software Performance Metrics (SPM) measure the effectiveness of the implemented Channelization Algorithm. These metrics include the following:

- Processing Time and Maximum Arrival Rate
- Memory Utilization and Memory per Request

End-to-end Delay and Maximum Arrival Rate

The end-to-end delay is defined as the amount of time required for a request to be processed and the corresponding channels defined. We define the maximum arrival rate as the maximum steady-state arrival rate of information requests that can be handled without error. In the absence of internal queues within the software, the maximum arrival rate is the inverse of the end-to-end delay.

Memory Utilization and Memory per Request

The memory utilization is defined as the amount of memory consumed during the execution of a channelization algorithm. For example, memory may be consumed due to internal hash tables and data caching used to reduce the end-to-end delay.

The memory per request is simply the memory consumption divided by the number of requests in the system. This provides a first order approximation to the memory requirements necessary to support a particular number of requests.

4.5 ALGORITHM DEFINITIONS AND SUMMARY CHARACTERISTICS

The following sections define the algorithms explored as part of ChannelTech and summarize many of the key results. Additional details for the algorithms and their properties can be found in the papers contained in Appendix B.

4.5.1 Trivial Algorithms (No Aggregation and Full Aggregation)

Three factors impact channelization algorithms: the set of subscription channels; the assignment of sources to channels; and the number of channels constructed over the network. Correspondingly, there are three trivial channelization algorithms defined below.

- No Aggregation (Receiver): Make a separate channel for each user containing precisely the information the individual user wants. Each user subscribes solely to his unique channel while each source feeds the channels that intersect its contents. This method may overwhelm the information system by sending many information products repeatedly, but it eliminates the unwanted information received by the users. Destination Transfer Agents will not have to filter out any information that is received. Rather the data that is received will be exactly what a user has asked for. The number of subscribed channels is always one. This approach is roughly equivalent to having separate information stove-pipes for each user.
- No Aggregation (Source): Make a separate channel for each source containing precisely the information that the source has. Each user subscribes to the channels that intersect their information interests. In this case, no information is sent redundantly but a user may receive excess information. Source Transfer Agents may have to transmit the same information item on multiple Channels, when two or more

users request the same type of data. The number of sources feeding a particular channel is always one. This approach is closer to the broadcast model where particular predefined channels carry news, sports, weather, and other interesting content. Users select the content that is most relevant to their needs.

- Full Aggregation: Make one single channel containing all the information products wanted by every user and have every user subscribe to that channel. This may overwhelm some users with unwanted information, but again avoids any redundant transmissions. In this case, the number of channels is always one.

In each case, the complexity of the algorithm is independent of the number of previous requests.

4.5.2 Direct (or Full) Decomposition

This approach is the basis for the analysis described in the introduction of this section. Specifically, the set of information requests is decomposed into its smallest set of disjoint subsets that cover the user's requests. Then, there is precisely one channel for each desired disjoint information set and these are singlecast or multicast to the users who are interested in them. This method has the advantage of minimizing both the amount of unwanted information and total information sent.

Unfortunately, as noted previously, the number of channels grows exponentially with the number of requests and is therefore not practical for large numbers of user requests. However, as described in *Utility-Based Management of Information Dissemination* [2], it is the basis for all orthogonal channelization algorithms and the fundamental construct for many of the mathematical formulations.

4.5.3 Containment Algorithms

In order to make use of multicast technologies (in an effort to reduce the amount of redundant data transmissions) it is desirable to define Channels that satisfy more than one Information Request. One approach taken to achieve this goal is the development of the Containment Channelization Algorithm.

The containment algorithm is governed by the comparison of the data regions associated with the existing channels and the current request. We refer to these data regions as Metadata Space Regions (MSRs) or Exchange Characteristics as they provide an abstract description of the information content. If the MSR of a received request is completely contained within the MSR of an existing channel, then the request is aggregated into the existing channel. Thus, the user's desired information is a subset of the information transmitted over the specified channel. Alternately, if the user's request covers a particular channel's definition, then the channel definition is updated to cover the broader scope. When no existing channel completely contains the request, a new channel is created.

The motivation for the containment algorithm is that users' requests are significantly similar and, therefore, can benefit from this reduction of channels without overloading any particular user with significant excess information. This assumption is motivated by the observation that WWW traffic satisfies a Zimpf distribution. On the other hand, when user's requests are uncorrelated and are not contained, then an unlimited number of channels may result similar to the trivial No-Aggregation (Receiver) algorithm. By the same token, any user submits a sufficiently general request for information, then the request may cover the entire set of channels and behave like the trivial full aggregation algorithm.

We can also define the "containment algorithm" using the matrix formulation presented earlier. If all the requested information products in row *i* are also requested in row *j*, then row *j* is said to contain row *i*. If any element of row *i* is one, while the corresponding element of row *j* is zero, then row *j* does not contain row *i*. Two equal rows will contain each other, except that one element of the set will not be marked as such. An all-zero row will be contained by any other row so it may be disregarded.

The savings in information sending depends on the user information requests and is delineated in detail in *Examining the Containment Algorithm* [3].

4.5.4 Reverse Containment Algorithms

Whereas the containment algorithm constructs channels using the definition that covers the other requests, the reverse containment algorithm constructs channels based on the channel that is being contained. Specifically, given a request for a subset of information that is currently supported on an existing channel, the reverse containment algorithm splits the channel into two disjoint channels: one for the desired subset and a second for the remaining information.

Channel reduction occurs when information requests can be composed from the set of existing channel definitions. Then, after the channels that are contained within the request are removed, the remainder is empty so no new channel is constructed. Thus, this algorithm is based on the assumption that user requests are commonly composed of a small set of aggregate components similar to the direct decomposition.

When this algorithm is tested on randomly constructed problems that have fewer information regions than users, the algorithm initially behaves by assigning one-channel-per-user, but eventually results in a small number of disjoint components. This is especially true, if the probability that a given user requests a given information region is either low or high. When there are many more information regions than there are users, the number of containers becomes small, and the reverse containment algorithm does nothing.

4.5.5 Clustering Algorithms

Further reductions in the number of channels may be accomplished by an algorithm (or a group of algorithms) called "clustering" algorithms. In this concept, when two or more user requests differ by little (perhaps as measured by the information density) each of them is assigned to a channel that covers them all, a channel made by unionizing the information regions in the nearby requests. Algorithms of this type produce results that depend on the precise ordering of comparisons and replacements. Clustering algorithms maintain the one-channel-per-user result. *Planned as a future task, exploration of this class of channelization algorithms was not explored in detail.*

5. STATIC CHANNEL SOFTWARE

5.1 INTRODUCTION

The ChannelTech software architecture is based on the BADD software architecture. The software developed under ChannelTech extends the functionality provided by BADD in key areas identified through the research. The primary objective of the ChannelTech software is to capture crucial measurements and experience to assist with the development of optimized Channelization Algorithms. As a result, this software is designated a prototype and has minimal operational requirements. With the exception of certain cross-contractor integration to support system-wide experimentation, there is no identified operational setting for this software.

5.2 EXPERIMENTAL SOFTWARE

5.2.1 Monte-Carlo Simulation (Cpcrib)

An experimental test program named Cpcrib was written to examine potential channelization algorithms. The program was written for a PC using MS Visual C++. While this flexible test program was not envisaged as deliverable software and is not a part of the prototype ChannelTech software, it is provided. An attached paper entitled *Experimental Software Description* [6] gives details of the software's construction and use. In addition, the program code is reasonably well documented internally.

5.2.2 State-Space Analysis (RIMEStateSpace)

Evaluation of channelization algorithms based on the state-space representation described in Section 2.3 was accomplished using an experimental Unix test program named RIMEStateSpace. This program:

1. Calculates state-space transitions and saves them to a file;
2. Calculates statistics for a given probability distribution of information requests;
3. Generates data for plots as a function of time.

The key control parameters are listed below. Additional options can be listed using the help option (-h or -H).

Table 3-1 - Key Control Parameters for RIMEStateSpace Program

-a <algorithm>	Channelization algorithm to use. Available choices are none (for no aggregation), single (for full aggregation), containment, and direct.
-d <distribution>	The distribution of information requests. Supported types include: • Uniform0: Each subregion of the decomposition has the same probability of being selected. Selecting no regions is permitted. • Uniform1: Same as Uniform0 except each request

	must select at least one region.
-r <number>	The number of subregions after the decomposition. This specifies the size of the bit vector described in Section 2.3. Numbers larger than 5 typically result in a prohibitive number of states to track.
-i <number>	The number of iterations to perform. When combined with the various output options (not listed), key statistics can be generated as a function of the number of requests in the system.

In addition, the program can dump a file that describes the state-space given a particular number of regions. Subsequent runs of the program can load this file and thereby reduce the time required to process results. Alternatively, users may construct these files manually to represent unique algorithms.

The state-space file is a text file consisting of two columns. The first column is a list of the channels' definitions and the added information request definition separated by colons. The second column is reduced list of channel definitions based on the algorithm. Both columns are sorted in numerical order with redundancies removed. For example, some of the valid state transitions from Figure 2.3 of Section 2.3 are

- 6:24: 6:24: Given one channel sending regions 2 and 3 (i.e., 110 or 6) and a request for regions 4 and 5 (i.e., 11000 or 24), the result is two channels with the same definitions. The same holds if the original channel had definition 24 and the request had definition 6.
- 6:21:24: 23:24: Given two channels with definitions 6 and 24, respectively and an input request with definition of 21, the resulting state is two channels with definitions 23 and 24.

5.3 STATIC CHANNEL OPERATIONAL SOFTWARE

The initial ChannelTech software supports 2 of the Channelization Algorithms previously discussed, the Trivial Algorithm and the Containment Algorithm. Additional Channelization Algorithms may be easily integrated into the ChannelTech Software by adhering to a well-defined algorithm interface.

The software CD is submitted under separate cover and includes documentation.

6. RESULTS AND DISCUSSION

This section investigates the differences and similarities of the various algorithms and their implementations using the metrics defined in Section 2.4. Additional details about the containment algorithm can be found in [3][5]. Table 4-1 summarizes the theoretical and experimental results in terms of the Channelization Analysis Metrics (CAM) defined in Section 2.4.

Table 4-1 - Comparison of CAM Values

CAM		No Aggregation (Rcv)	Full Aggregation	Containment	Direct Decomp.
Channel Count (n_c)	Min	n_r	1	1	n_r
	Max	n_r	1	n_r	$2^{n_r}-1$
	Mean	n_r	1	see Fig. 4-1	(*)
	Std	0	0	see Fig. 4-1	(*)
Redundant Gain ($\tilde{G}_r$)	Max	1	0	1	0
	Mean	1	0	see Fig. 4-2	0
	Std	0	0	see Fig. 4-2	0
Subscription Count (n_s)	Min	1	1	1	1
	Max	1	1	1	$2^{n_r}-1$
	Mean	1	1	1	(*)
	Std	0	0	0	(*)

(*) Neither closed-form expression nor sample experimental results available.

6.1 CHANNEL ALGORITHM MEASUREMENT RESULTS

Validation of the algorithm characteristics was accomplished through a series of experiments and benchmarks performed using the simulation and operational software described in Section 3. This section discusses these results.

6.1.1 Uniform Distribution

A set of 18 trials was performed whereby information requests were generated uniformly over a set of exchange characterization attributes and submitted at a fixed rate between five to fifteen seconds per request. Each exchange characteristic was randomly generated over three data types (i.e., image, text, video); twenty-six keywords; two access methods (i.e., HTTP or FTP); and optionally stationary targets and/or ATO products. For each request, the user's information needs were selected randomly from these values and submitted to the system. In addition, all the attributes with the exception of keywords, could be skipped implying that the user did not care about the value for that attribute.

Figure 4-1 shows the channel count as a function of the number of information requests for the containment algorithm using both the experimental testbed and the Monte-Carlo simulation. Analogous to the trivial receiver-oriented trivial algorithm, the number of channels grows linearly with a slope of slightly below one channel/request for small numbers of requests. That is, each request results in a new channel since the likelihood of overlap is small. For example, for the 18 trials, there was an average of 9.37 channels after 10 requests. For large numbers of requests, the containment algorithm approaches a small set of orthogonal channels that cover the other requests. In this case, the limiting value is 26 channels corresponding to each of the letters in the alphabet. After 1000 requests, the experiment yielded an average of 32 channels over the 18 trials. In the Monte-Carlo simulations, we see that the expected number of channels is 26.2 after 2000 information requests. Correspondingly, the standard deviation of the mean decreases as well. For example, the experimental standard deviation at 1000 information requests is 5.86 compared to a standard deviation of 10.37 at curve's peak.

In producing a set of orthogonal channels as the number of requests grows, the containment algorithm reduces the amount of redundant data transmitted. This is shown in Figure 4-2. Like the previous plots of the channel count, the number of channels that overlap initially rises as the algorithm behaves similar to the trivial receiver-oriented algorithm, peaks, and then decreases as it settles into a set of disparate channels. However, the peak of these curves in both theory and experiment are shifted from the peaks in the number of channels. Note also that, when compared to the overlap in the requests, the approximate redundant gain decreases exponentially with the number of information requests. The reason is that while the channel overlap decreases exponentially, the request overlap increases exponentially. Fitting the theoretical values to an exponential curve reveals the approximation

$$\tilde{G}_r = (0.1072)e^{-0.0047 n_r}$$

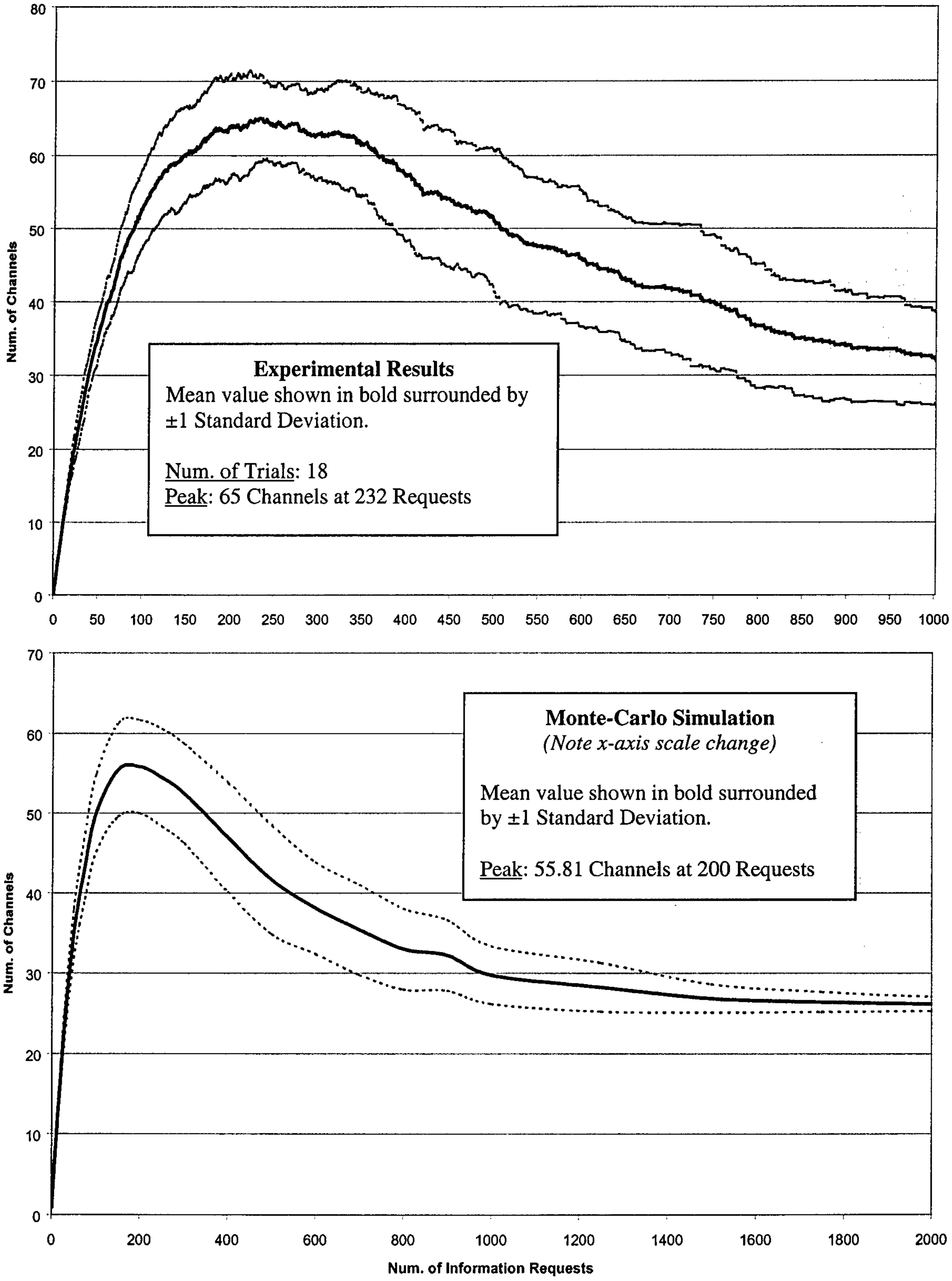
with a coefficient of determination of 0.9879.

6.1.2 Lantica Scenario

As part of the PA&I investigation, an operational scenario was constructed designed to demonstrate the functionality of the AIC components. Designed to operate against the limited tagging associated with the connection-oriented AIC layer, this scenario was limited to nine different requests for information.[3] The scenario also included information about the utility of the requests along with the timing of the various submissions. From this data, the Lantica experiment was performed against the testbed and the results are shown in Figure 4-3.

As can be seen, the system quickly settled onto the five orthogonal channels with occasional drops to four channels when there were no active requests for Warning data. Furthermore, early during the simulation when the number of requests are small, the number of channels equals the number of requests as noted in the previous section.

[3] Of these requests, only one (Intelligence : All Source) covered any of the other requests. Specifically, the four specific types of Intelligence data were assumed to be subsets of the All Source requests. When taken as a whole, this implied that there were only five orthogonal requests (i.e., 9 - 4).



Experimental Results

Mean value shown in bold surrounded by ±1 Standard Deviation.

Num. of Trials: 18

Peak: 65 Channels at 232 Requests

Monte-Carlo Simulation

(Note x-axis scale change)

Mean value shown in bold surrounded by ±1 Standard Deviation.

Peak: 55.81 Channels at 200 Requests

Figure 4-1: Number of Channels for the Containment Algorithm

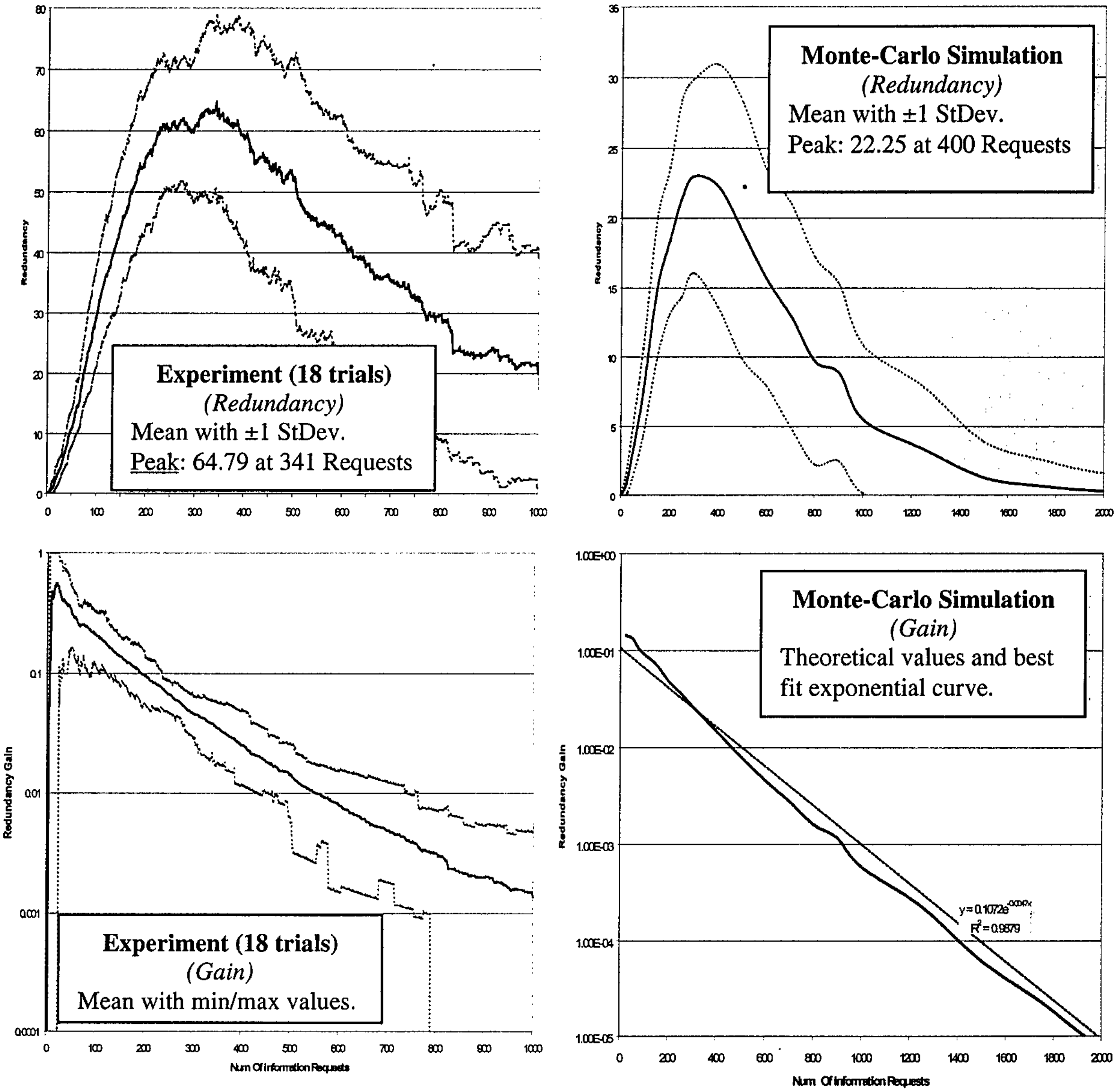
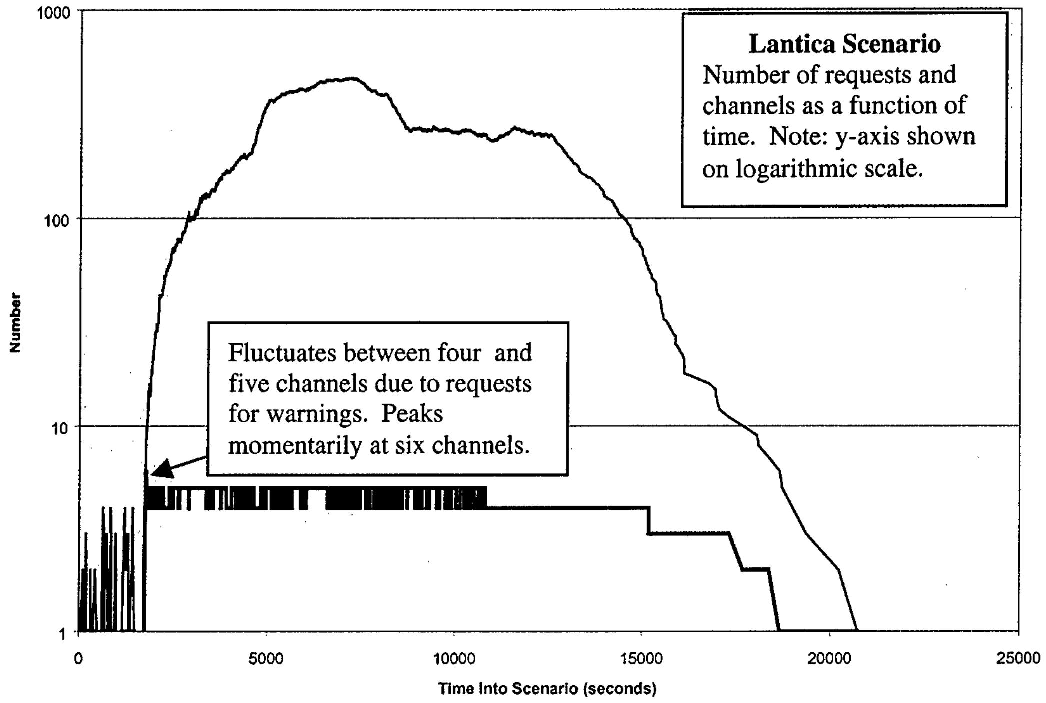


Figure 4-2: Experimental and Theoretical Redundancy Measurements



Lantica Scenario
Number of requests and channels as a function of time. Note: y-axis shown on logarithmic scale.

Fluctuates between four and five channels due to requests for warnings. Peaks momentarily at six channels.

Figure 4-3: Number of Channels as a Function of Time for Lantica

•

6.2 SOFTWARE PERFORMANCE MEASUREMENT RESULTS

The processing time and memory utilized is summarized in Table 4-2 and shown in Figure 4-4. These results are based on a set of uniformly distributed information requests as described in the previous section.

Table 4-2 - SPM Metrics for Operational Software

SPM		No Aggregation (Rcv)	Containment
Processing Time	Mean	383 msec	764 msec
	Min	32 msec	27 msec
	Max	94.671 msec	44.070 msec
	Std	4.188 msec	2.255 msec
Memory per Request	Mean	45 kbytes	
	Min	15 kbytes	
	Std	79 kbytes	

During the trials, it was noted that the processing delay between adding an information request and removing an information request was asymmetric. That is, removing requests took more time than adding requests. One reason is that if the removed information request was the request that contained multiple others, then its removal could result in splitting a single channel into smaller, non-contained channels. This calculation required more comparisons that adding an information request.

Another observation is that the time processing time for the no aggregation algorithm is about half the time required to process the containment algorithm. This result is not surprising in that no aggregation does not require any comparisons between various requests.

Examination of the memory consumed as a function of the number of profiles in the system was approximately linear. This allows for easy estimation of the amount of memory required to support a particular peak number of information requests. In addition, the memory consumed is significantly less than similar measurements taken on version 4 of the BADD IDM Core. In all, the improved framework reduced the memory from approximately 250K bytes/profile to 45 kbytes/profile. Note that the times reported here are slightly longer than those required by the previous version of the BADD Core. This is due, in part, to the more flexible architecture as well as the additional support for the containment algorithm that was not present in BADD.

7. CONCLUSIONS

Early analysis of the channelization problem reveals both the complexity and early characteristics of the problem. In its simplest form, the channelization problem grows exponentially with the number of requests in the system. Furthermore, non-linearities inherent in the problem preclude the use of simply iterative approaches that result in local minimums.

Despite these difficulties, the potential benefits of channelization can be noted in both the theoretical and experimental analysis performed to date. For example, substantial reductions in the number of channels hosted within the network as well as the load imposed due to redundant transmissions can be noted using the simple containment algorithm. This report explored a number of possible algorithms and developed an infrastructure for analyzing their performance. This included trivial algorithms, containment, reverse containment, and the direct (or full) decomposition.

Additional benefits may be obtained by recognizing many of the characteristics of the surrounding environment. The Lantica scenario demonstrated a military scenario where the commonality between user's information needs was very high. As a result, only four channels were necessary to support the user's needs. More generally, information requests are often Zipf-like and opportunities for optimizing algorithms by leveraging this need to be explored.

Many of the software artifacts developed under the ChannelTech project have been transitioned to other projects and shall be included in a future DII/COE GCCS release as part of the IDMDom segment. This report also provides a central repository for the technical artifacts to support future investigations necessary to provide next-generation information management services.

8. REFERENCES

1) S. Zabele, "Channelization Techniques for Data Dissemination," TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, April 1998.
2) B. DeCleene, "Utility-Based Management of Information Dissemination", TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, August 1999.
3) G. Matchett, "Examining the Containment Algorithm", TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, August 1999.
4) B. DeCleene, S. Griffin, G. Matchett, R. Niejadlik, "Real-time Information Management Environment (RIME)", To appear in Proceeding of SPIE AeroSense - Digitization of the Battlespace V, April 2000.
5) G. Matchett, "Channelization", TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, December 1999.
6) G. Matchett, "Experimental Software Description", TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, December 1999.

9. ACRONYMS

AIC	Utility-Enabled Networking (formally Agile Information Control)
AICE	Agile Information Control Environment
AEI	Information-Enabled Networking (formally AICE Enhanced Information Management)
AITS/JPO	Advanced Information Technology Systems Joint Program Office
BADD	Battlefield Awareness and Data Dissemination
BC2A	Bosnia Command and Control Augmentation
CAM	Channelization Algorithm Metric
DARPA	Defense Advanced Research Projects Agency
DII/COE	Defense Information Infrastructure Common Operational Environment
FACB	Functional Architecture Control Board
FTP	File Transport Protocol
GCCS	Global Command and Control System
HTTP	Hyper Text Transport Protocol
IDM	Information Dissemination Management
IP	Internet Protocol
IPM	Information Policy Management
ISR	Information Space Region
ISO	Information Systems Office
ITO	Information Technology Office
MSR	Metadata Space Regions
JPO	Joint Program Office
PA&I	Performance, Analysis & Integration
QOS	Quality of Service
RIME	Real-Time Information Management Environment
ROI	Region of Interest
SECB	System Engineering Control Board
SPM	Software Performance Metric
TBD	To Be Defined
TCP	Transport Communication Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language

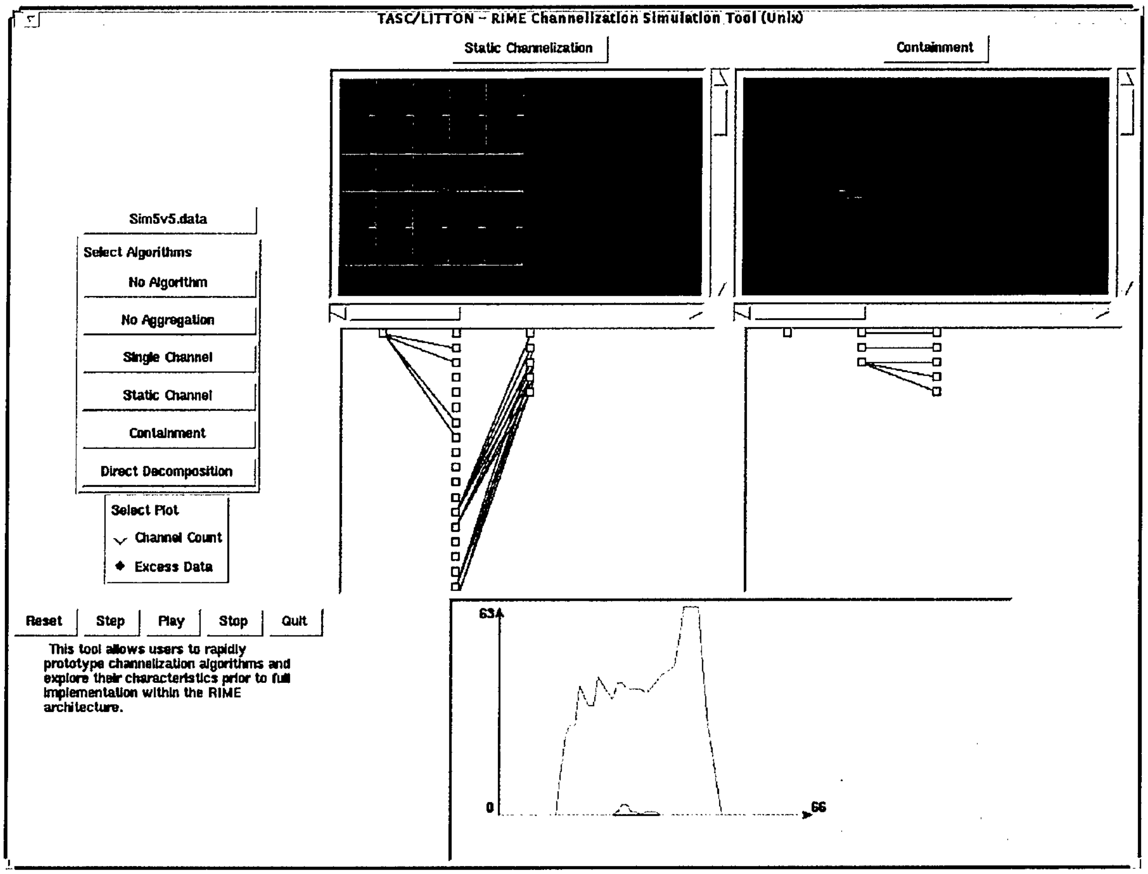
APPENDIX A - RIME SOFTWARE USER MANUALS

This appendix contains details on the execution of the primary software products generated as part of ChannelTech.

As the RIME Editor and RIME Server are based upon the BADD software architecture, only the differences in usage between the two are discussed in this appendix. For full details about the user interface and server installation and usage, please see the latest BADD/IDM DII/COE documentation.

A.1 RIME SIMMULATION TOOL

Complementing the research, RIMESimTool allows users to prototype channelization algorithms and visually explore the results prior to implementation within the RIME Server. A screenshot of the tool is shown in Figure A-1. This tool is authored in Perl/Tk ver 5.0 and has been demonstrated on both Solaris and NT platforms. Definitions of the supported channelization algorithms can be found in the corresponding research section.



This tool allows users to rapidly prototype channelization algorithms and explore their characteristics prior to full implementation within the RIME architecture.

Figure A-1: RIME Simulation Tool Interface

A.1.1 User Interface

The left side of the tool contains controls for selecting the type of simulation, algorithms, and information to display. The elements of the control panel from top to bottom are as follows:

- **Top button**: Clicking this button opens a file browser allowing the user to select the simulation to execute. Format and construction of simulations are described in Section A.1.2.

- **Select Algorithm Frame**: Within this frame is a list of all of the algorithms supported. Selecting one of these buttons causes alternate panels to use the chosen algorithm. The name of the algorithm assigned to each visualization panel is shown at the top of the panel. In Figure A-1, static channelization and containment were selected respectively.
- **Select Plot Frame**: Within this frame, users can choose the type of information that they would like plotted during the simulation in the plot panel.
- **Reset Button**: This button stops the simulation and resets it to the beginning.
- **Step Button**: Executes a single command within the simulation file.
- **Play Button**: Causes the simulation file to be executed.
- **Stop Button**: Stops a simulation that is being played.
- **Quit Button**: Quits the application
- **Information Panel**: This region displays various information as the user selects different parts of the visualization panels or plots. When a component of the visualization panel is selected, this region shows the definition of the selected item. When a plot is selected, the statistics associated with the plot are shown.

The visualization panels allow users to see the state of the system as a function of time. Two views are provided as part of this display. The top view (shown in black) displays information requests in red, channels in green, and advertisements in blue. The outline represents the region of interest for each of the respective elements. For example, we can see that the region of interest for the advertisement in the left visualization panel (blue) intersects four channels.

The lower region of the visualization panel shows the connections between the various definitions. From left-to-right, the columns of boxes correspond to advertisements, channels, and information requests respectively. For example, we can see that the single advertisement associated with the static channelization algorithm intersects four channels. Therefore, four lines emanate from the box in the advertisement column to four channel boxes in the middle column. In the same fashion, we can see which channels the information requests have subscribed to.

In both panels, these various components can be selected by clicking the mouse on the desired item. When selected, the item will become bold and any relevant additional information will be displayed in the information panel.

A.1.2 Simulation File Format

Simulation files consist of a series of commands that can be executed by the tool during operation. Any number of blank lines or comments (denoted with #) can be included in the file for readability.

Regions of interest (ROI) are specified as ordered pairs of integers on a fixed rectangular grid. These ordered pairs are represented on the visualization panel as complex polygons. Semi-colons may separate multiple ordered pairs to denote complex regions. Furthermore, a dash between two values means to select all points between the specified values as well as the end-points. An example is shown in the Figure A-2.

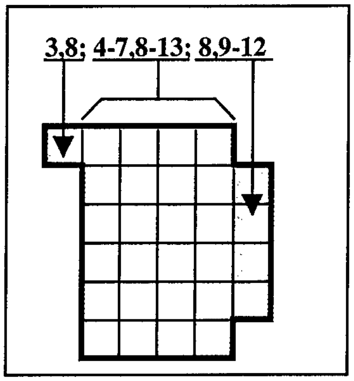


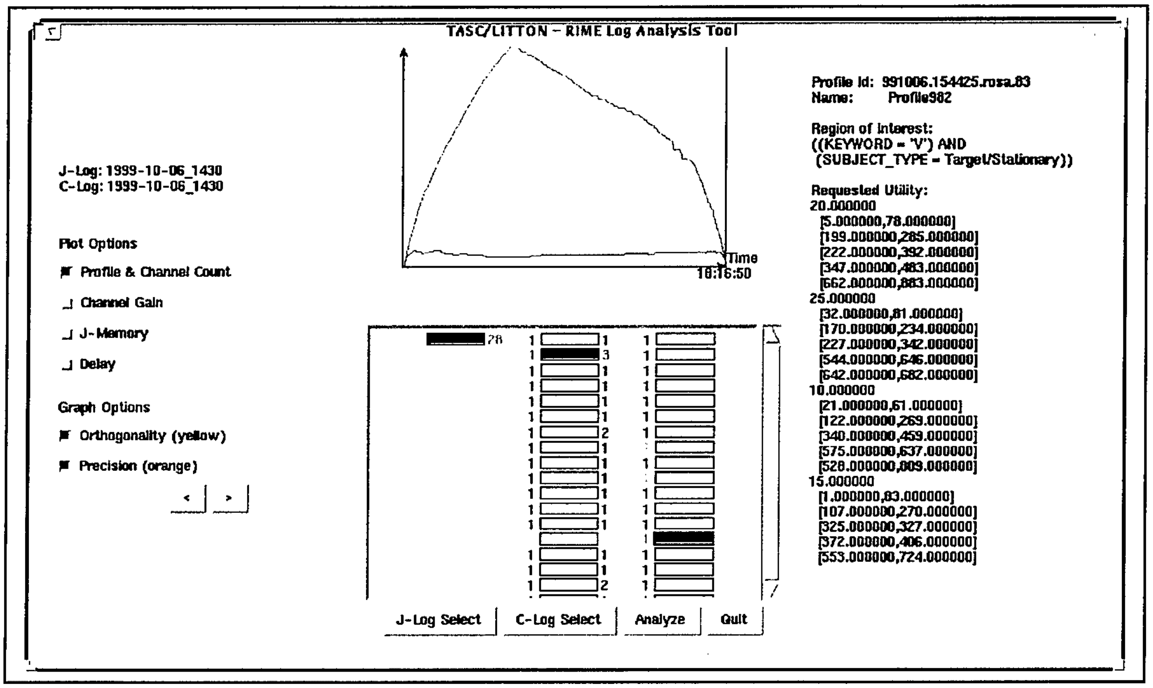
Figure A-2: Regions of Interest are Specified as Ordered Pairs of Integers on a Fixed Rectangular Grid

The commands that are supported are

- **Advert: <name> <ROI>**
  Add or modify an advertisement corresponding to the name. The region of interest for the advertisement corresponds to the ROI.
- **Profile: <name> <ROI>**
  Add or modify an information request corresponding to the name. The region of interest for the information request corresponds to the ROI.
- **Profile: <name> remove**
  Remove the specified information request.
- **Profile: <name> random <type> <parameters>**
  Generate a random information request of the given name. The type field defines the method to use in generating the information request. The parameters depend upon the type. Only 'uniform' distribution is currently supported.
- **Profile: <name> move x,y**
  Moves the information request by the corresponding x,y amount.
- **Channel: <name> <ROI>**
  Add or modify a channel corresponding to the name. The region of interest for the channel corresponds to the ROI.

A.2 RIME LOG ANALYSIS TOOL

Figure A-3 shows the RIMELogAnalysis tool. This tool allows users to examine the state and performance of the RIME Server's channelization algorithms.



TASC/LITTON – RIME Log Analysis Tool

J-Log: 1999-10-06_1430
C-Log: 1999-10-06_1430

Plot Options

- ☒ Profile & Channel Count
- □ Channel Gain
- □ J-Memory
- □ Delay

Graph Options

- ☒ Orthogonality (yellow)
- ☒ Precision (orange)

```
Profile Id:  991006.154425.rosa.83
Name:        Profile982

Region of Interest:
((KEYWORD = 'V') AND
 (SUBJECT_TYPE = Target/Stationary))

Requested Utility:
20.000000
  [5.000000,78.000000]
  [199.000000,285.000000]
  [222.000000,392.000000]
  [347.000000,483.000000]
  [662.000000,883.000000]
25.000000
  [32.000000,81.000000]
  [170.000000,234.000000]
  [227.000000,342.000000]
  [544.000000,646.000000]
  [642.000000,682.000000]
10.000000
  [21.000000,61.000000]
  [122.000000,269.000000]
  [340.000000,459.000000]
  [575.000000,637.000000]
  [528.000000,809.000000]
15.000000
  [1.000000,83.000000]
  [107.000000,270.000000]
  [325.000000,327.000000]
  [372.000000,406.000000]
  [553.000000,724.000000]
```

Figure A-3: RIME Log Analysis Tool Interface

The tool is divided into four key regions: display controls, plot, state graph, and information. These are located on the far-left, middle-top, middle-bottom, and far-right respectively. In addition, there are four buttons at the bottom of the tool. The "J-Log Select" and "C-Log Select" buttons allow the user to navigate to the two log files necessary for the analysis. The selected log files are shown on the left. Pushing "Analyze" causes the log files to be reexamined. *Hence, while an experiment is in progress, the log files can be specified once and reanalyzed whenever the user wants to refresh his display to capture the latest system-state.* The Quit button exits the tool.

Display Controls

The display controls on the far-left allow the user to select which plots to show in the plotting area and any special information in the state graph. The supported plots are:

- Profile & Channel Count: Displays the number of information requests (i.e., profiles) and corresponding number of channels as a function of time. This option is selected in Figure A-3.
- Channel Gain: The channel gain is defined as the ratio of the number of channels divided by the number of requests.

- J-Memory: The amount of memory consumed by the channelization server as a function of time.
- Delay: The delay from information request submission to when the final channel definitions and agent instructions are generated.

The state graphs shows the relationships between the various source advertisements, channels, and information requests at a particular instance. Described in greater detail below, the following is a list of support graph options:

- Orthogonality: This option allows the user to identify channels with information content that overlap. For example, a channel sending Korean information and a channel sending news would overlap as both would carry Korean news reports. Overlapping channels are shown in yellow.
- Precision: This option allows the user to identify which subscribers to a particular channel are receiving excess information. For example, a user interested in financial news subscribing to a generic news channel would also receive information about sports and weather that are outside of his interests. Users receiving excess information are shown in orange.

Below the state graph options, there are two buttons that allow the user to step the graph forward and backward in time. This also results in a time-bar on the plot diagram that represents the current position in time. For example, in Figure A-3, the time-bar is currently at 18:60:50 with the corresponding state diagram for that time shown at the bottom.

Plot Region

The plot region (shown in the top-center) shows all of the plots selected in the display controls as a function of time. Right-mouse-clicking on any of the plots causes detailed information about its statistics to be printed in the information area on the far-right. The selected plot will become bold to distinguish it from the other plots. *In addition, the time-bar can be dragged to any particular time.*

State Graph

The state-graph (shown in the bottom-center) displays the source advertisements, channels, and information requests at the time specified by the time-line. This is organized into three columns. The column on the left is a list of the various source advertisements. The number next to each box corresponds to the number of channels that this source is feeding. For example, in Figure A-3, there is precisely one advertisement that is feeding 29 channels.

The middle column is a list of the channels. Each channel is being fed by any number of sources and has any number of subscribes. The number of sources feeding the channel is shown to the left of the box while the number of subscribing information requests is shown on the right.

The right column is a list of the information requests. In this case the number of channels that the request is subscribed to is shown on the left of the box.

Like the plots, any box can be selected with a single mouse click and the resulting information is shown in the information region on the far-right. The selected box is outlined in bold. In Figure A-3, the information request shown in red is currently selected and its corresponding details shown on the right. This information request (named Profile982) is interested in stationary targets with the keyword "V."

Double-clicking any box within the state graph highlights itself in blue and the boxes in the other columns that are connected in red or orange.[4] Figure A-3 second channel from the top has been double-clicked. As can be seen, the single advertisement is the source for this channel and the three-colored information requests are all subscribed to this channel.

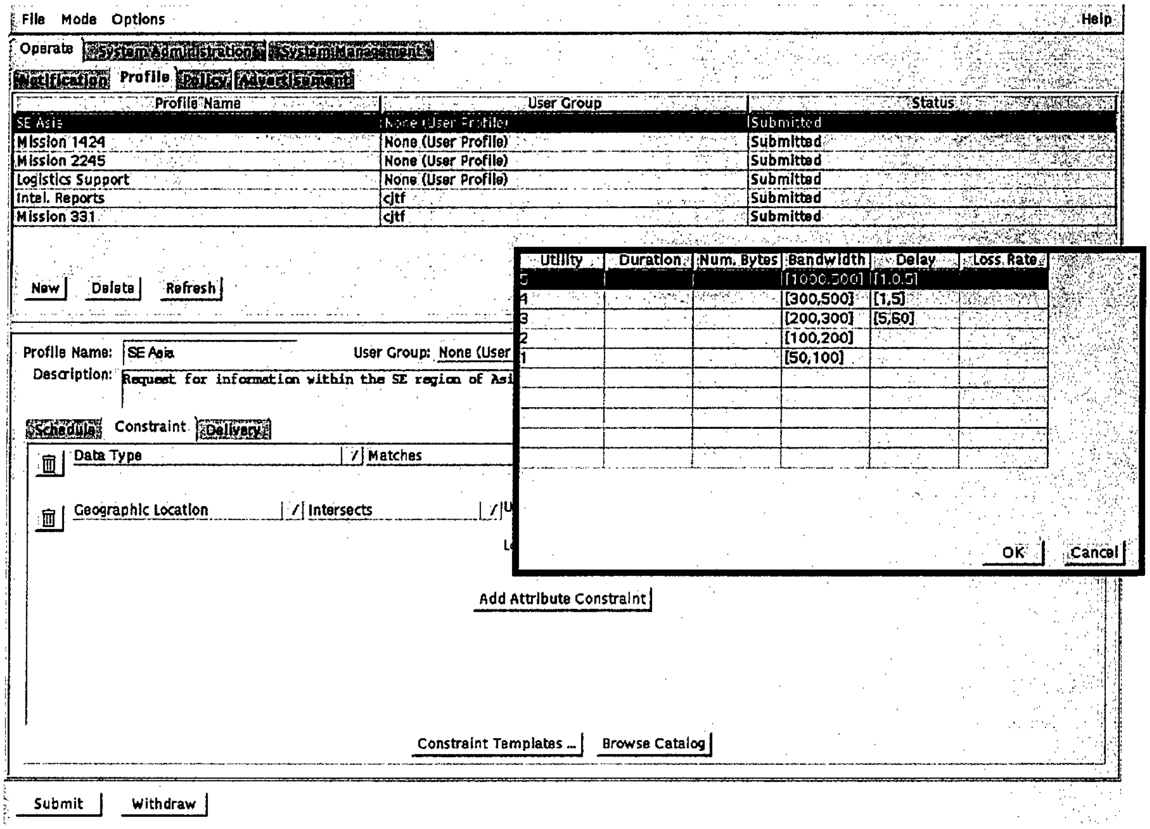
Information Panel

As described previously, the information panel shows key data corresponding to selected boxes in the state-graph or plots.

[4] Boxes will highlight orange only if the "Precision" option is turned on.

A.3 RIME EDITOR V1.0

Figure A-4 shows the RIME Editor tool. The tool is similar to the DII/COE IDM Editor (IDMEdt) but also allows users to specify utility functions using the small panel shown within the figure below. Details on the BADD editor usage can be found in [BADD99u]. This table is constructed as a list of regions that have constant utility value. Each row of the table corresponds to a specific set of QoS parameters that correspond to a particular utility value. Each element of the utility table is active so that selecting it will result in a pop-up enabling the user to choose from available valid values.



Profile Name	User Group	Status
SE Asia	None (User Profile)	Submitted
Mission 1424	None (User Profile)	Submitted
Mission 2245	None (User Profile)	Submitted
Logistics Support	None (User Profile)	Submitted
Intel. Reports	cjtf	Submitted
Mission 331	cjtf	Submitted

Profile Name: SE Asia User Group: None (User

Description: Request for information within the SE region of Asi

Data Type Matches

Geographic Location Intersects

Utility	Duration	Num. Bytes	Bandwidth	Delay	Loss Rate
5			[1000.500]	[1.0.5]	
4			[300,500]	[1,5]	
3			[200,300]	[5,60]	
2			[100,200]		
1			[50,100]		

Figure A-4: RIME Editor Interface

A.4 RIME SERVER V1.0

The RIME Server developed under AICE is based on the BADD Phase 2 Core Server version 4.3.6. Details on its usage can be found in [BADD99s]. This section discusses the user issues that are unique to the RIME Server.

The behavior of the RIME Server is controlled in part by the configuration parameters specified in the file *idmjServer.prp*. During AICE, additional properties were added to allow for the seamless selection of the channelization algorithms. The following is an excerpt from this file along with additional comments. Comments are denoted with #.

```
#
# Property controlling the Channelization Algorithm used by Delivery.
# This algorithm specifies how channels are constructed from
# the profile definitions.  Only two algorithms are currently supplied.
#
# For no aggregation algorithm uncomment out the following line:
# com.tasc.badd.de.NoChannelizationAlgorithm
#
# For the containment algorithm (CCA), uncomment the following:
# com.tasc.badd.de.CCA
#
com.tasc.badd.de.channelizationAlgorithmName =
  com.tasc.badd.de.NoChannelizationAlgorithm

#
# Property controlling the Channel Selection Algorithm used by Delivery.
# This defines how receivers select their channel subscription.  Currently
# there are two algorithms.
#
# Used with the No Channelization Algorithm above.  This algorithm selects
# channels whose originating id matches the id provided in the profile.
# com.tasc.badd.de.NoChannelizationSelectionAlgorithm
#
# Used with the Containment Algorithm.  Selection is based on identifying
# all channels that intersect the provided profile's exchange characteristics.
# com.tasc.badd.de.ContainmentSelectionAlgorithm
#
com.tasc.badd.de.selectionAlgorithmName =
  com.tasc.badd.de.NoChannelizationSelectionAlgorithm

#
# Property controlling the comms component implementation.
# There are three "MetaNet" interfaces that are provided as
# part of the RIME Server.  They are…
# com.tasc.badd.de.AICInterface : Interface to AICE/AIC services
# com.tasc.badd.de.MetanetInterface : Interface to BADD MetaNet
# com.tasc.badd.de.WaatsInterface : Interface to IDM WAATs services
#
#
com.tasc.badd.de.commsComponentName = com.tasc.badd.de.AICInterface
```

APPENDIX B - PUBLISHED PAPERS, ARTICLES AND THESIS

The following are a set of papers and articles generated during the ChannelTech project.

- B. DeCleene, "Utility-Based Management of Information Dissemination", TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, August 1999.
- B. DeCleene, S. Griffin, G. Matchett, R. Niejadlik, "Real-time Information Management Environment (RIME)", To appear in Proceeding of SPIE AeroSense - Digitization of the Battlespace V, April 2000.
- G. Matchett, "Examining the Containment Algorithm", TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, August 1999.
- G. Matchett, "Channelization", TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, December 1999.
- G. Matchett, "Experimental Software Description", TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading, MA 01867, December 1999.

Real-time Information Management Environment (RIME)

Brian DeCleene*, Sean Griffin, Gary Matchett, Richard Niejadlik

Litton/TASC, Reading, MA 01867

ABSTRACT

Whereas data mining and exploitation improve the quality and quantity of information available to the user, there remains a mission requirement to assist the end-user in managing the access to this information and ensuring that the appropriate information is delivered to the right user in time to make decisions and take action.

This paper discusses TASC's federated architecture to next-generation information management, contrasts the approach against emerging technologies, and quantifies the performance gains. This architecture and implementation, known as Real-time Information Management Environment (RIME), is based on two key concepts: information utility and content-based channelization. The introduction of utility allows users to express the importance and delivery requirements of their information needs in the context of their mission. Rather than competing for resources on a first-come/first-served basis, the infrastructure employs these utility functions to dynamically react to unanticipated loading by optimizing the delivered information utility. Furthermore, commander's resource policies shape these functions to ensure that resources are allocated according to military doctrine. Using information about the desired content, channelization identifies opportunities to aggregate users onto shared channels reducing redundant transmissions. Hence, channelization increases the information throughput of the system and balances sender/receiver processing load.

Keywords: Information Management, Information Dissemination, policy, profile, channelization, resource optimization

1. INTRODUCTION

Achieving mission-critical situational awareness through information dominance requires timely and efficient delivery of information in response to changing conditions. While improved sensor collection platforms and data processing techniques such as data mining enrich the depth and breadth of available information, they do little to ensure that the right information is delivered to the right users in a timely fashion to ensure overall mission success. Rather, most systems allocate and manage their resources on a first-come/first-served basis with users competing for the available resources. As a result, information is distributed according to the time of the user's request rather than the importance of the information that is being conveyed. Although recent developments in Quality-of-Service (QoS) guaranteed networks help alleviate this problem, these solutions do not consider how the connections are being used in determining how information is distributed to whom.

During a crisis, the problem is further aggravated by the observation that the necessary information not only exceeds the available capacity of the network resources but is often generated by unanticipated sources. Hence, static doctrine that dictates a fixed chain of information dissemination can introduce costly delays and potentially misrepresent situations to the end user. By emphasizing the management of information over the management of connections, it is possible to allow the system to route information to the required users without a-priori knowledge of the source of the information.

Information (or content-based) processing extends the traditional channel request/management services by exploiting details about the information that is intended to flow over the connection. Some of the services that are available under content-based processing include:

- Smart Warfighter Profiling: Commonly, Warfighters have a clear understanding of their information requirements. Unfortunately, mapping these needs to network resources, such as bandwidth, is difficult since the end-user has only partial knowledge about what is available at the producer and how it will be disseminated. Under the RIME architecture, the Warfighter's profile only specifies information requirements and the importance of that data to the mission success -- no a-priori end-user determination of necessary bandwidth is required. The RIME system, in concert with the producer, determines the network resources required to satisfy the Warfighter's profile and allocates these accordingly.

* Correspondence: Email btdecleene@tasc.com; Telephone: (781)205-7243; Fax: (781)942-9507

- Commander's Policy: When network resources are oversubscribed, the Commander needs to be able to define rules to ensure that resource allocations are consistent with overarching military doctrine and mission priority. Like profiling, these rules are expressed in terms of information requirements versus network connectivity. For example, a Commander may specify that data corresponding to a particular area of interest is mission-critical over other information requests. RIME uses this information to adjust the allocation of system resources accordingly.
- Improved Information Throughput: Besides mapping information needs and policies into system resources, additional opportunities for increasing the dissemination of mission-critical information are available in the RIME architecture. For example, Warfighters often have common information needs. By identifying these shared needs and multicasting the data to all of the users simultaneously, the system can reduce the number of redundant transmissions. Similarly, under traditional information paradigms, users rely on a small set of statically defined repositories for their information needs. Unfortunately, this means that alternative sources of data are often missed or discovered late. Furthermore, users commonly identify changes and new data by polling at various intervals. This results in additional delays in accessing mission-critical information. By combining the Warfighter's information interests with the producer's summaries of available content, RIME identifies all sources of value and ensures that the desired content is routed to the user accordingly.

This paper formalizes the notions behind these concepts and discusses how they are implemented within the RIME architecture. In the next section, the elements of a Warfighter profile are formalized along with their interaction with producer information to allocate system resources. Then, Section 3 defines the notion of Commander's policy and provides examples. Increasing information throughput through channelization is discussed in Section 4. In Section 5, the RIME implementation is discussed in detail. Finally, Section 6 compares this research to current systems.

2. WARFIGHTER PROFILES - METADATA SPACE REGIONS AND UTILITY FUNCTIONS

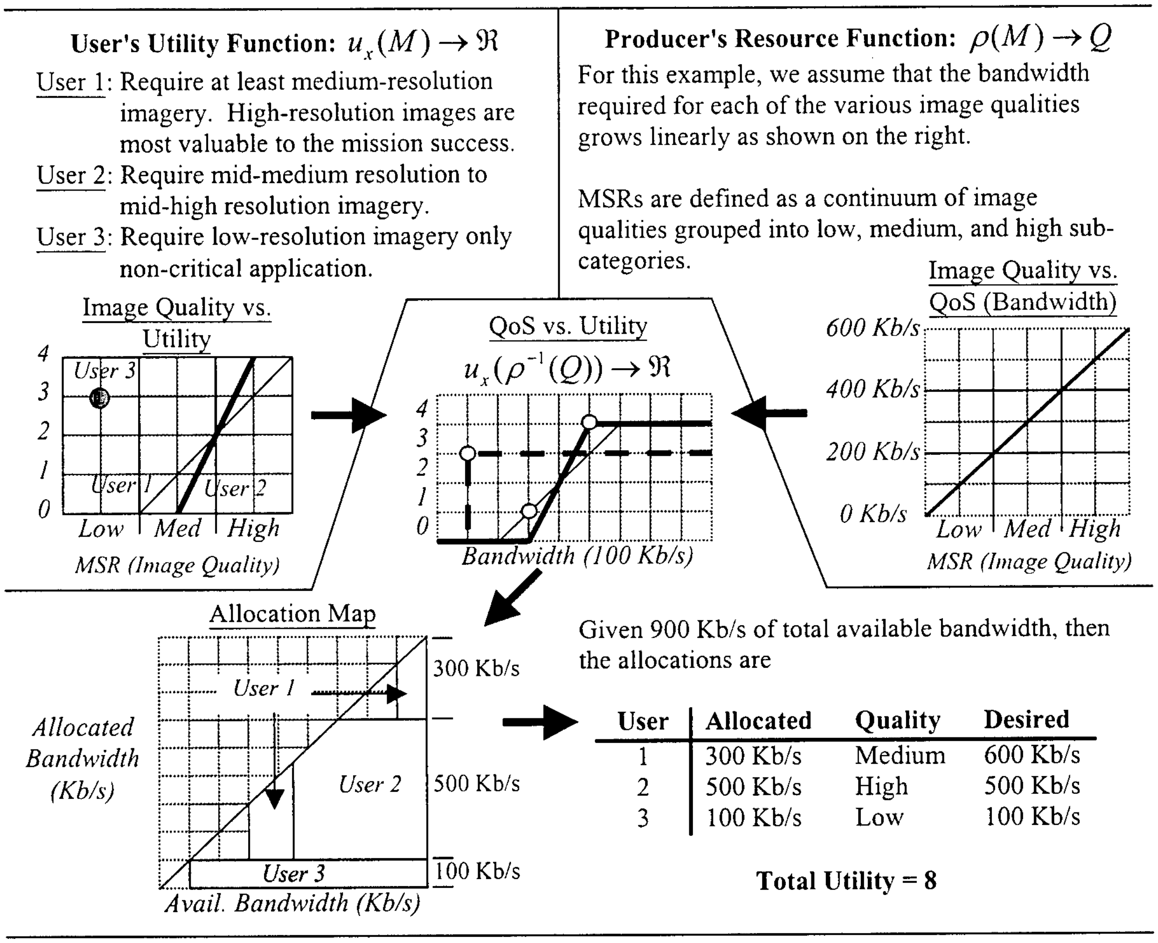
Central to the architecture, the Warfighter's profile consists of two main parts: an expression of his information interests and the corresponding value of that information to the success of his mission. A *Metadata Space Region (MSR)* defines a user's information interests by specifying the subset of the set of all possible products that are relevant to the user. For example, a valid MSR is "intelligence traffic within central Bosnia." Rather than specifying specific products for dissemination, the MSR defines a criterion that is used to determine which current and future products the user desires.

It is important to note that an MSR is an abstract representation of a set. Consequently, set operations such as intersection and union can be applied to various users' MSRs. For example, the intersection between user A's MSR "intelligence traffic within central Bosnia" and user B's MSR "signal intelligence traffic within Europe" is simply "signal intelligence traffic in Bosnia." That is, any signal intelligence product associated with Bosnia is of interest to both users.

The second main part of a profile, utility, measures the "value" of the requested information to the success of the mission. Consequently, the information management system may trade-off system resources against various requests in order to maximize the utility of the delivered information. Formally, we begin by defining a *utility function* as a mapping of MSRs to the non-negative real numbers (i.e., the utility function for user x's information needs is $u_x(M) \to \Re^+$) such that the larger the number, the more critical the information is to the success of the mission. Some properties of this function based on the MSR's set operations include:

- Information that has no value to the mission has a utility of zero — delivering this information does not increase the mission's ability to succeed. Information that is not within the range of the utility function should not be delivered to the user at all.
- Expanding the information delivered increases the utility of the information. Mathematically, this implies if $M_1 \subseteq M_2$, then $u_x(M_1) \le u_x(M_2)$.[1]
- If a request for information is divided into parts, then the utility of the original request equals the total utility of the parts. Specifically, if $M_1 \cap M_2 = \varnothing$, then $u_x(M_1) + u_x(M_2) = u_x(M_1 \cup M_2)$.[2]

[1] Note that this is a statement of the value of the information and not the processing load required use the information. Rather, the issue of information overload and end-host processing is addressed later during resource optimization.



User's Utility Function: $u_x(M) \to \Re$

User 1: Require at least medium-resolution imagery. High-resolution images are most valuable to the mission success.

User 2: Require mid-medium resolution to mid-high resolution imagery.

User 3: Require low-resolution imagery only non-critical application.

Producer's Resource Function: $\rho(M) \to Q$

For this example, we assume that the bandwidth required for each of the various image qualities grows linearly as shown on the right.

MSRs are defined as a continuum of image qualities grouped into low, medium, and high sub-categories.

Given 900 Kb/s of total available bandwidth, then the allocations are

User	Allocated	Quality	Desired
1	300 Kb/s	Medium	600 Kb/s
2	500 Kb/s	High	500 Kb/s
3	100 Kb/s	Low	100 Kb/s

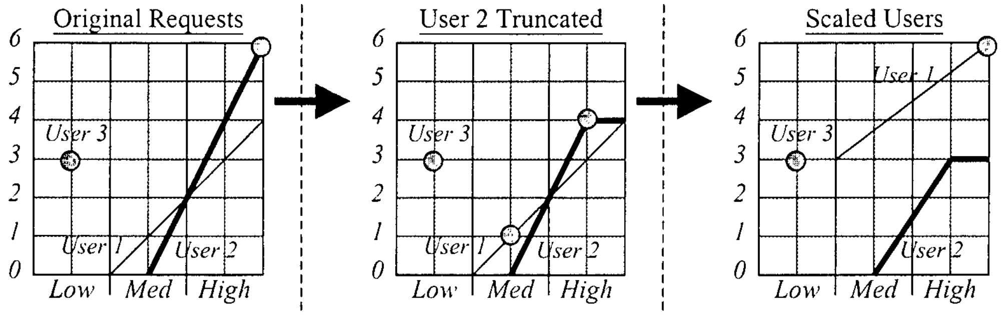
Total Utility = 8

Figure 1: RIME Mapping of User Needs to Network Resources

By defining the utility function over the MSR, the user does not need to know a-priori the networking resources (e.g., bandwidth) that is required to deliver the information. Instead, the information producer defines a *resource mapping* of MSRs to resource requirements. Mathematically, $\rho(M) \to Q$ where Q is the quality-of-service (QoS) required to support the specified MSR. Through this partnership between the user's utility function and the producer's resource mapping, users manage information rather than networks, and the RIME architecture optimizes the performance to maximize the value of information delivered.

To illustrate how these concepts interact within the RIME architecture, consider three users with varying information needs as shown in Figure 1. On the left, each user has specified his individual information requirements and the value of the information to his mission. For example, User 2 (designated by the bold line) prefers high quality imagery but medium quality imagery will also suffice for his mission. Low-resolution imagery is not sufficient and therefore has no value to User 2. The right side shows the producer's bandwidth requirements as a function of the information. Combining these two functions together, the system identifies the value of the system resources to each of the users (shown in the middle of the figure). Next, the system assigns resources to each of the users to optimize the total delivered utility as a function of the available bandwidth. The allocation map (shown on the lower-left) shows the resource allocations for each user. For example, with only 900 Kb/s of available bandwidth, User 1 receives less bandwidth than desired in order to allocate it to the more critical needs of Users 2 and 3. If instead, only 400 Kb/s were available, then User 2 would not receive any bandwidth.

[2] A more flexible generalization of this property is based on the concept of *conditional utility* where the utility is a function of the other information delivered. This allows users to account for the fact that some information has less value unless it is delivered with other information.



User	Allocated
1	none
2	600 Kb/s
3	100 Kb/s

***Total Utility* = 9**

User	Allocated
1	300 Kb/s
2	500 Kb/s
3	100 Kb/s

***Total Utility* = 8**

User	Allocated
1	600 Kb/s
2	none
3	100 Kb/s

***Total Utility* = 9**

Figure 2: Example of Policy Applied to Utility Functions and Resulting Allocations

3. COMMANDER'S POLICY - UTILITY MODIFYING FUNCTIONS

A problem with utility-based information management is that requests can be preempted by arbitrarily increasing the utility of one request to ensure that all other requests are blocked. This behavior may result in resource allocations that are counter to the overall mission success. Successful operational information management must take into account the fact that not all missions (and therefore users) are equally important at all times. For example, a request for information from a special operations team may take precedence over a similar request from a logistics group. RIME addresses these issues through policies that adjust user requirements according to military doctrine.

A policy is defined as a rule that tailors a particular individual or group's information requests. Upon submission, the user's utility function is modified into a new utility function by all policies applicable to that user. We shall denote this policy function as $\alpha_i \circ u_x \equiv \alpha_i(u_x) \to u_y$. Consequently, policy may modify the allocation of resources in the system. Before illustrating, let us note the intuitive properties of these policy functions and their corresponding mathematical implications.

- Once a policy is applied to a utility function, then that utility function is compliant with the policy. That is, reapplying the rule a second time does not change the utility function. Mathematically, this implies that $\alpha_i \circ \alpha_i \circ u_x = \alpha_i \circ u_x$.
- Policies may be aggregated into master policies. That is, policies are associative — $(\alpha_i \circ \alpha_j) \circ \alpha_k \circ u_x = \alpha_i \circ (\alpha_j \circ \alpha_k) \circ u_x$. For performance reasons, this observation allows the RIME architecture to pre-calculate aggregate policies for particularly dynamic users. Then, rather than repeatedly applying multiple policies to the utility function every time it changes, the aggregate policy can be immediately applied once. For this reason, an approach to pre-calculating aggregated policies is being considered as a technique for reducing the processing time required to resolve a request against policy.
- Not all policies are equal. For example, a policy of a CINC commander may take precedence over the policies of a battalion commander. As a result, policies are not necessarily commutative ($\alpha_i \circ \alpha_j \circ u_x \neq \alpha_j \circ \alpha_i \circ u_x$).

Given these observations, we can now recognize specific policy functions that comply with the mathematical formalism.

Truncating — The simplest form of policy is to restrict any user from exceeding a particular maximum utility. Thus, wherever the utility function exceeds a specified threshold, it is truncated to the maximum allowed value. For example, let us consider the two left-most plots in Figure 2. In this case, all of the utility functions are truncated to a maximum value of four, causing User 2's utility function to be cropped. Then, using the parameters from the previous example to optimize the delivered utility, the quality of information delivered to User 2 is reduced, allowing User 1 to be allocated 300 Kb/s. Not only does this definition satisfy the properties outlined previously, we note that truncating policies are commutative. Truncating loses the details of the utility function for everything being truncated. In this case, although the user has explicitly identified that very high-resolution imagery is more valuable than high-resolution imagery, the truncated utility gives them the same value to the mission's success. As a result, no resources will generally be allocated to provide the higher quality of service since it consumes resources without increasing the total delivered utility.

Scaling (or Ranking) — Unlike truncating, scaling policies preserve the shape of the original utility function. This allows certain users to receive preferential treatment without losing the relative value of the information to the Warfigher. Mathematically, if $u_{\min}$ and $u_{\max}$ are the minimum and maximum values of the utility function u_x, then the scaling policy into the interval [a,b] is given by

$$S_{a,b} \circ u_x = (b-a)\left[\frac{u_x - u_{\min}}{u_{\max} - u_{\min}}\right] + a$$

For example, reconsider Figure 2. On the right a scaling policy has been applied giving User 1 priority over Users 2 and 3. Specifically, User 1 is rescaled to the interval [3,6] while Users 2 and 3 are rescaled to the interval [0,3]. As a result, User 1 receives additional bandwidth to meet his needs while User 2 receives no bandwidth. Again, it is easy to show that scaling policies adhere to the properties listed previously. Unlike truncating, however, scaling is invertible.

Credit-based Policy — An example of a more sophisticated policy function is based on a credit-model where each user has a finite number of utility units that can be assigned to the various information types. As a result, a user can only increase the utility of a particular type of information by reducing the utility of some other information. Unlike truncating and scaling, this approach ensures that each user does not assign all information the highest allowable utility value. Mathematically, normalizing everything to k units, a credit-based policy is simply

$$I_k \circ u_x = {}^{k \cdot u_x}\Big/_{\int_M u_x}$$

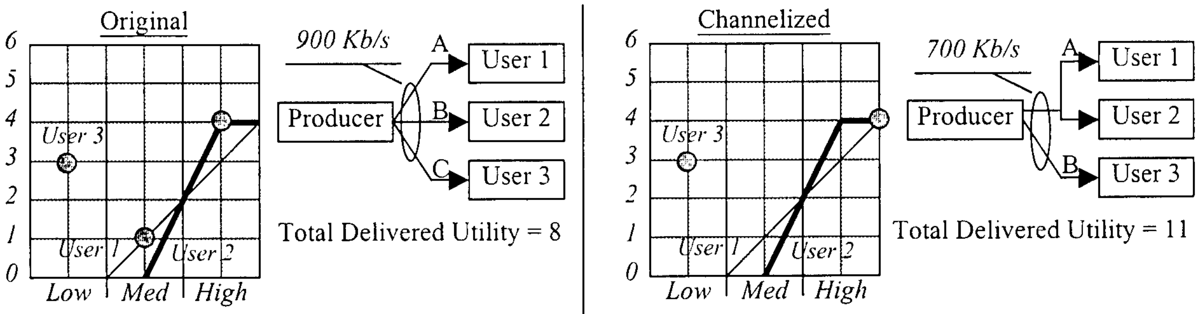
While truncation policies commute, generally this is not the case. Changing the order in which scaling or credit-based policies are applied changes the final policy-corrected utility function. As a result, policy requires a precedence rule that determines the order for applying policy. In the military, the command hierarchy provides the primary rule for determining precedence of policies as well as identifying which policies apply to which users. Specifically, each commander and warfighter in the RIME system are associated with a mission and unit within the command hierarchy. Each commander may designate the set of subordinate units that is subject to the policy. The policy is then applied to all users in those units. Policies from superior commanders take precedence over policies from their subordinates. While more complicated interrelationships are admissible within the RIME framework, this simple approach has proven sufficient for current military needs.

Another observation is that setting a utility function to zero does not guarantee that the user does not receive the information. Rather, a zero utility only ensures that solutions that deliver extra data are not favored. As we will see in the next section, delivering information with zero utility to a particular user on a shared channel may actually increase the total system utility. To explicitly preclude certain solutions during resource optimization, RIME supports *access policies*. A sample access policy may prevent a user from using a particular satellite link or receiving data associated with a specific area of interest by changing the range of the utility function.

4. CHANNELIZATION

In the previous sections, policy function tailored each user's information request by modifying the utility function. Then, the system allocated resources to maximize the utility of the delivered data. However, the RIME architecture extends the optimization a step further by comparing requests from different users and constructing shared data channels. By coordinating the placement of transmit- and receive-filters on shared multicast sessions, the system reduces the number of duplicate transmissions while simultaneously increasing the overall system utility. Based on studies of information requests on the Web, it has been shown that information requests commonly follows a Zipf-like distribution — a few products are desired by a large number of users and a small number of products are required by only a few users. [1][2] This observation provides the motivation for channel optimization.

The benefits of channelization can be illustrated by reconsidering the example described in Section 2. Without channelization, the system allocates the 900 Kb/s to three different connections as shown on the left of Figure 3. However, it



Chnl	Content	Allocation	Subscribers
A	Med.	300 Kb/s	User 1
B	High	500 Kb/s	User 2
C	Low	100 Kb/s	User 3

Chnl	Content	Allocation	Subscribers
A	Very High	600 Kb/s	User 1, User 2
B	Low	100 Kb/s	User 3

Figure 3: Channel Aggregation Increases Information Utility

is clear that both users 1 and 2 are interested in receiving high-quality imagery. Therefore, by constructing a shared channel that contains the content of interest for these users, we are able to increase the overall utility as shown on the right.

RIME constructs channels by examining the overlap in MSRs for the users' utility functions and constructing a set of channel utility functions. Examples of channelization approaches are illustrated in Figure 4. As may be seen, subscribing to a certain subset of channels can satisfy each user's information needs. Formally, given a set of users' utility functions, $u_1,\ldots,u_N$, channelization constructs a new set of channel utility functions, $v_1,\ldots,v_M$, such that for all points within each user's utility function, there exists at least one channel satisfying $u_i(m) \leq v_j(m)$. If there is no duplication in the content on any of the channels (i.e., their MSRs do not overlap), we say that the channels are *orthogonal*. Similarly, if only data that a user is interested in (i.e., whose utility is non-zero) is delivered as part of his subscription, then we say that the channels are *exact*. On the producer side, if only information that at least one user has explicitly requested is transmitted on each channel, then the channels are *precise*. Alternative formulations including a state-space model and 0-1 matrix optimization problem can be found in [3].

The design of a channelization algorithm for a particular environment requires numerous considerations. For example, many networks and end-hosts have limitations on the number of connections that they can simultaneously support. Many approaches, such as the direct-decomposition, can rapidly run out of available connections. Aggregating into fewer channels

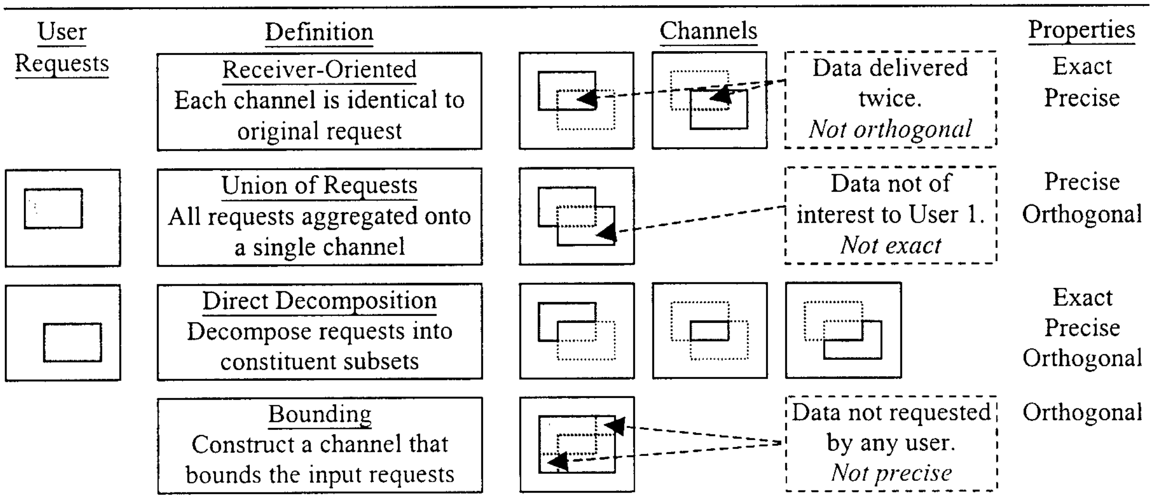

User Requests
Definition
Channels
Properties
Receiver-Oriented
Each channel is identical to original request
Data delivered twice.
Not orthogonal
Exact
Precise
Union of Requests
All requests aggregated onto a single channel
Data not of interest to User 1.
Not exact
Precise
Orthogonal
Direct Decomposition
Decompose requests into constituent subsets
Exact
Precise
Orthogonal
Bounding
Construct a channel that bounds the input requests
Data not requested by any user.
Not precise
Orthogonal

Figure 4: Sample Channelization Algorithms and Corresponding Properties

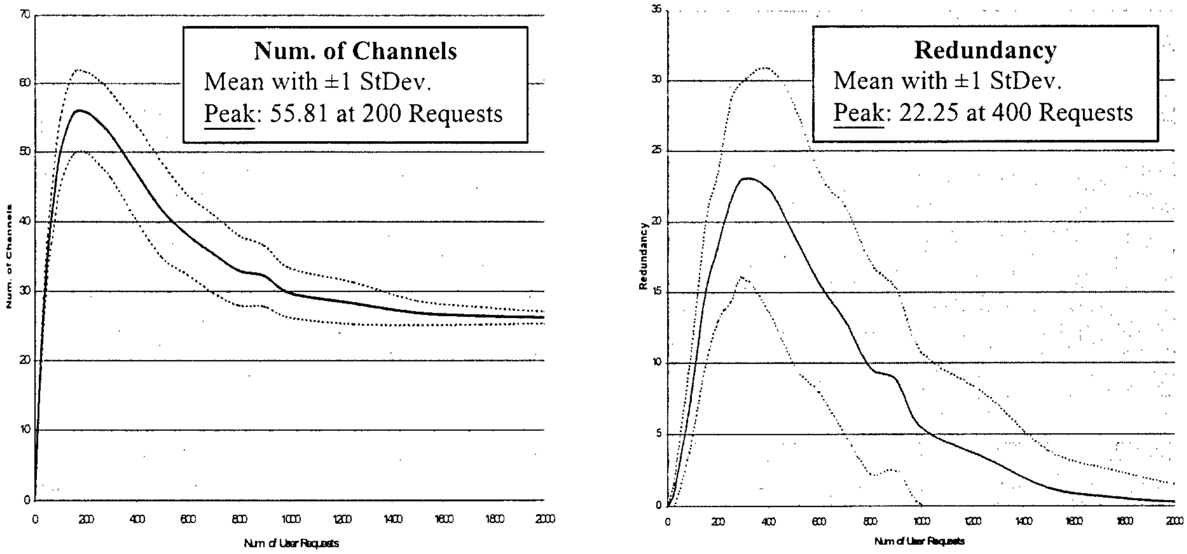


Figure 5: Performance Characteristics of Containment Algorithm for Uniformly Distributed Requests

results in either excess data being transmitted or received. The optimal channel set must identify the set of channels that balances the load on the network against the end-host filtering requirements. Additional difficulties include:

- Complexity: Given N profiles, there are at most 2^N-1 unique subsets that can be constructed from intersections of a given user profile set (or its complement) with each of the other user sets. The number of ways that these subsets can be combined into M channels grows as a Stirling number of the second kind. Hence, performing an exhaustive examination of channel definitions to identify the optimal solution is double-exponentional in complexity, O(M^{2^N}), which is prohibitive for all but a small number of users. [4]
- Non-Iterative: It can be shown that iterative solutions that build on the previous solution converge to local minimums that do not necessarily coincide with the globally optimal solution.

One straightforward algorithm that has been investigated extensively as part of RIME is the containment algorithm. Under containment, the range (i.e., MSR) of each request is compared against the ranges of each of the channels. If any channel's MSR fully contains the MSR of the new request, then the request is aggregated onto that channel and its utility functions added. Similarly, any channels that are within the MSR of the request are aggregated into a broader channel.

An advantage to the containment algorithm is that, if the user requests are independent, then it substantially reduces the number of channels supported within the network. This is shown in on the left in Figure 5. These plots were based on a Monte-Carlo simulation with approximately 30 correlated attribute values and independent user requests. As may be seen, initially the algorithm grows linearly, similar to the receiver-oriented algorithm defined in Figure 4. This is due to the fact that initially the likelihood that a request is contained by another request is small; therefore each request is assigned a unique channel with the same definition. As the number of requests increases, aggregation begins to occur. In the limit, the algorithm settles into clusters that encompass various user groups. In this case, the limiting set of channels was 25.

The containment algorithm not only reduces the number of channels required to support the user community, but it also reduces the amount of data being transmitted redundantly. The plot on the right of Figure 5 shows the number of channels with definitions that overlap with other channels (i.e., the number of non-orthogonal channels). When the number of requests is small, many of the channels overlap without being contained. Yet, as the number of requests increases, the containment algorithm converges to a set of orthogonal channels corresponding to clusters of users. Thus, each channel carries information for a different user community.

5. RIME OPERATIONAL ARCHITECTURE

The services outlined in the previous sections have been implemented as part of the RIME software architecture. To facilitate integration with legacy systems, the RIME implementation operates as a value-added layer on top of the existing

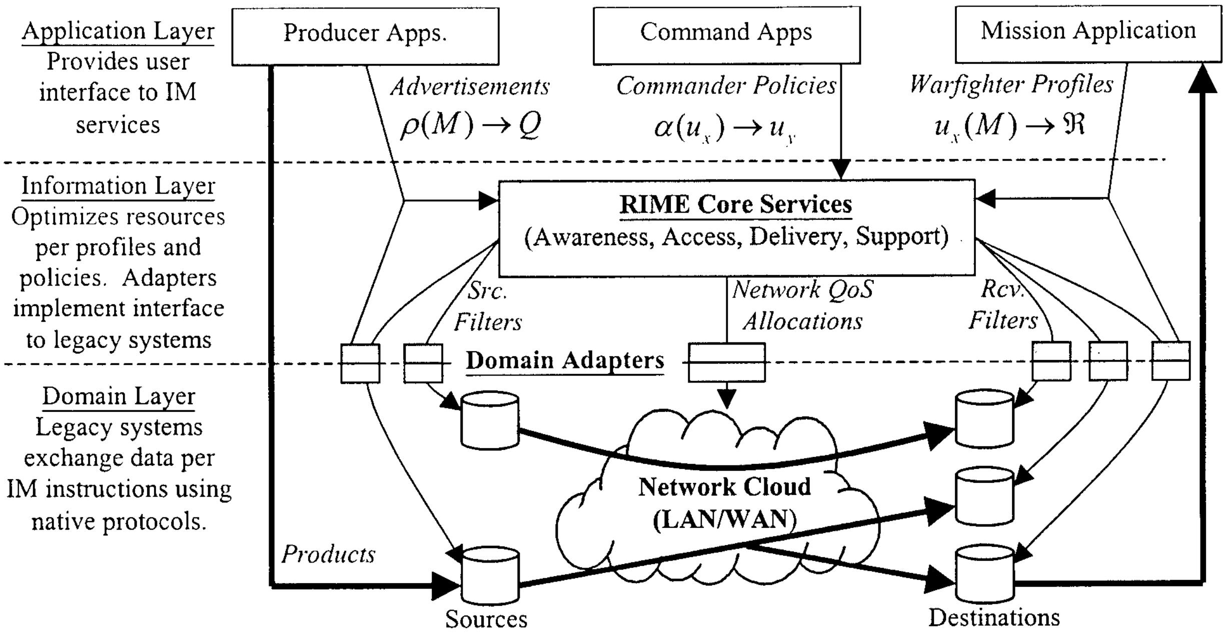


Figure 6: RIME Implementation Leverages Domain-Specific Services within a Common Framework

domain-specific legacy services as illustrated in Figure 6. By isolating the domain services from the information management services, the legacy flow of data from the sources on the left to the destinations on the right remains unchanged. With the addition of information management, the destination of information is not necessarily identical to the original source enabling an image library may deliver data onto a directory tree on the destination.

Accomplishing this, the RIME Core server performs Warfighter Profile and Commander Policy management outlined previously. After applying policy and optimizing the resource allocations, the Core service generates instructions to the domain adapters (also called transfer agents). On the source side, filters based on the channelization define what information should be transmitted on each channel. Expressed as an MSR, the source-side adapters identify which information satisfies the criteria and trigger its transmission accordingly. Similarly, on the destination side, receive filters tell the consumer applications what channels to subscribe. In this case, the MSR is used to potentially filter out data that the user is not interested in but was sent if the channel subscription was not exact. The Core services also inform the network services of any quality-of-service allocations derived from the producer's resource map.

The RIME architecture provides a federated approach to information management. As may be seen, each layer operates autonomously of the higher layers with the information layer coordinating actions between the various domains. Domain Commercial-Of-The-Shelf (COTS) products optimized for particular domains, such as Netscape Compass WEB server, may then be incorporated into the domain layer without requiring any modifications to their software. Under the DARPA/ISO Battlefield Awareness and Data Dissemination (BADD) program, numerous third-party capabilities have been integrated to support a variety of data types including file, WEB, Predator video, and imagery. Because the information layer does not interfere with the flow of data (shown in bold) in the domain layer, information services can be introduced without interrupting existing domain-specific services — an important requirement for deployment. Special transport protocols such as image tiling standards thus retain their functionality. Although it is not shown in the diagram, the architecture supports the notion of smaller sub-domains, with each Core responsible for its own area and only coordinating with the other Cores when necessary. This capability helps improve system scalability.

Consistent with the federated approach, TASC's RIME software only maintains summary informational awareness of the lower layers through the resource maps and thereby avoids data replication. Construction of product and summary advertisements is accomplished either manually or through automated discovery. Active domain adapters on the source-side can automatically update the information provided to the RIME Core services by examining the flow of data in their associated repository. For example, using a mix of government and COTS products, completeness of metadata collected for approximately 50,000 products was increased from 11% to 90% using automated discovery tools over manual examination.

TASC's RIME software suite consists of three key object-oriented segments corresponding to the Core services. the domain adapter toolkits. and the user interfaces. Consistent with the Joint Technical Architecture designation for the use of either the Distributed Computing Environment standards or the Common Object Request Broker Architecture (CORBA) family of specifications for distributed computing services, the RIME Core software uses CORBA for its inter-process coordination. Yet, RIME is also undergoing work to support customized control protocols tailored to particular government communications services. Table 1 below provides a quick summary of the key RIME software capabilities.

6. COMPARISON TO EXISTING SYSTEMS

The need for information management services to improve the quality and manage the quantity of information delivered to users over fixed network resources has resulted in numerous commercial and government solutions to this problem. Two areas of particular interest are the development of new WEB services and satellite broadcast management.

Within the commercial world, the exponential growth of information content on the WEB has resulted in numerous information management solutions designed to improve the access and delivery of HTML pages to users. Roughly, these solutions fall into the following categories:

- WEB-based Content Discover and Indexing (e.g., Spiders): These tools extract and catalog WEB pages to provide easy access to WEB information. Typically, these tools process the text content and keywords embedded into the various HTML pages. This information is then indexed according to a set of keywords enabling regular expression searches over an ensemble of repositories from a single database. Solutions for content discovery are directly applicable in identifying the relevant data per the information management layer's instructions and generating source summary.
- WEB-based Push Technologies and Notification: Companies such as Tibco (in partnership with Yahoo) and EntryPoint/PointCast, allow users to subscribe to URLs or channels that are proactively pushed to the user. [5] Besides keeping the user informed of changes in advance, push systems can use multicasting to simultaneously distribute the same

Table 1: TASC Real-time Information Management Environment (RIME) Software Fact Sheet

Core Services	Provides smart-push profiling capabilities and commander's policy services. Also, enables producers to advertise repository content. Channelization algorithm plug-ins enable tailored resource optimization based on information needs. Administration functions are available through user interface application as well as WEB-browser.
Editor Services	Provides access to Core capabilities according to assigned role. Display is configurable by local user. Template library exists for common profiles and policies. Notification is integrated with WEB browser.
Domain Adapter Toolkits	JAVA-based transfer agent toolkit facilitates third-party integration with commercial products. Hides server-to-server communication details.
Server-to-Server Communications	CORBA/IDL. Also supports custom protocols for constrained networks.
Military Standards	Submitted for DII/COE Level 7 certification.
Commercial Standards	Utilizes IDL specification for external interfaces. Supports XML for logging and notification services. Compliant with JAVA event services.
Platforms	Core services operate on Unix/Solaris. User interface to services available for both Unix and Windows systems.
Mapping Tool	Configurable. Currently supports SpatialX.
Schema	Run-time configurable. RIME Editor automatically reconfigures according to current schema and allows for local tailoring of terminology.
Security Services	Strong-encryption of password-protected accounts and supports SSL. Each user is provided access limited to the services corresponding to their assigned role in the system such as Warfighter, Commander, or Administrator.

advertisements is accomplished either manually or through automated discovery. Active domain adapters on the source-side can automatically update the information provided to the RIME Core services by examining the flow of data in their associated repository. For example, using a mix of government and COTS products, completeness of metadata collected for approximately 50,000 products was increased from 11% to 90% using automated discovery tools over manual examination.

TASC's RIME software suite consists of three key object-oriented segments corresponding to the Core services. the domain adapter toolkits. and the user interfaces. Consistent with the Joint Technical Architecture designation for the use of either the Distributed Computing Environment standards or the Common Object Request Broker Architecture (CORBA) family of specifications for distributed computing services, the RIME Core software uses CORBA for its inter-process coordination. Yet, RIME is also undergoing work to support customized control protocols tailored to particular government communications services. Table 1 below provides a quick summary of the key RIME software capabilities.

6. COMPARISON TO EXISTING SYSTEMS

The need for information management services to improve the quality and manage the quantity of information delivered to users over fixed network resources has resulted in numerous commercial and government solutions to this problem. Two areas of particular interest are the development of new WEB services and satellite broadcast management.

Within the commercial world, the exponential growth of information content on the WEB has resulted in numerous information management solutions designed to improve the access and delivery of HTML pages to users. Roughly, these solutions fall into the following categories:

- WEB-based Content Discover and Indexing (e.g., Spiders): These tools extract and catalog WEB pages to provide easy access to WEB information. Typically, these tools process the text content and keywords embedded into the various HTML pages. This information is then indexed according to a set of keywords enabling regular expression searches over an ensemble of repositories from a single database. Solutions for content discovery are directly applicable in identifying the relevant data per the information management layer's instructions and generating source summary.
- WEB-based Push Technologies and Notification: Companies such as Tibco (in partnership with Yahoo) and EntryPoint/PointCast, allow users to subscribe to URLs or channels that are proactively pushed to the user. [5] Besides keeping the user informed of changes in advance, push systems can use multicasting to simultaneously distribute the same

Table 1: TASC Real-time Information Management Environment (RIME) Software Fact Sheet

Core Services	Provides smart-push profiling capabilities and commander's policy services. Also, enables producers to advertise repository content. Channelization algorithm plug-ins enable tailored resource optimization based on information needs. Administration functions are available through user interface application as well as WEB-browser.
Editor Services	Provides access to Core capabilities according to assigned role. Display is configurable by local user. Template library exists for common profiles and policies. Notification is integrated with WEB browser.
Domain Adapter Toolkits	JAVA-based transfer agent toolkit facilitates third-party integration with commercial products. Hides server-to-server communication details.
Server-to-Server Communications	CORBA/IDL. Also supports custom protocols for constrained networks.
Military Standards	Submitted for DII/COE Level 7 certification.
Commercial Standards	Utilizes IDL specification for external interfaces. Supports XML for logging and notification services. Compliant with JAVA event services.
Platforms	Core services operate on Unix/Solaris. User interface to services available for both Unix and Windows systems.
Mapping Tool	Configurable. Currently supports SpatialX.
Schema	Run-time configurable. RIME Editor automatically reconfigures according to current schema and allows for local tailoring of terminology.
Security Services	Strong-encryption of password-protected accounts and supports SSL. Each user is provided access limited to the services corresponding to their assigned role in the system such as Warfighter, Commander, or Administrator.

content to multiple users thereby saving network resources. Lacking the policy and channelization services outlined previously, and tailored for a particular community, these systems complement the RIME architecture by providing improved end-to-end throughput within the WEB domain.

- WEB-based Forward Staging (Proxy Servers): Forward staging allows users to access either pushed content or previously accessed information without having to reconnect over the network. The perception to the user is that he is fully connected to the Internet even though much of his access is local. When utilized as a receive-side domain adapter, proxy servers often address issues such as document organization and garbage collection — critical issues to the end-user. Examples include GrapeVine and Netscape Compass.

Compared to the RIME implementation, pure WEB solutions are typically limited in the semantics supported for expressing information needs and their ability to manage resource usage. For example, as most solutions rely on text-based indexing, it is difficult to accommodate non-text queries such as geographic polygons or temporal ranges. Along the same lines, since most Internet-based products compete for resources using the TPC/IP congestion algorithms, they cannot dynamically allocate network resources to favor one mission over another nor utilize policy to adjudicate between competing requests.

In the government, information management technologies have been evolving for some time. For example, many message-based systems allow users to define the relative importance of various classes of information. Then, messages are examined against these filters and assigned a corresponding priority. Deduplication and reordering ensure that flash and critical messages receive preferential treatment over routine traffic. While these systems operate well within their domain, they do not provide the policy services nor general utility-based resource management necessary to span multiple domains. As a result, improvements to the national systems to provide real-time seamless access to information are necessary.

DARPA began sponsoring research in information management technologies in the mid-1990s and demonstrated the concept as part of JWID 1995 and Bosnia Command & Control Augmentation (BC2A). Additional research was performed under the DARPA/ISO BADD Phase I and Phase II programs. BADD program developed and demonstrated a functional and software architecture demonstrating many of the information management services. This research was the basis for the RIME architecture. Under BADD, users expressed information requests using constant utility value (precedence). Access policy was applied to these requests, and the precedence value was potentially truncated by a single resource policy that specified the maximum allowed value for a particular mission. The validated profile was then channelized using the receiver-oriented algorithm described earlier. Information types such as file, WEB, streaming video, and database data were all managed by this system. Under the DARPA/ITO Agile Information Control Environment (AICE) program, these initial information management concepts were extended to perform real-time information management and improve resource management through the introduction of utility described in this paper.

7. CONCLUSION

The military is undergoing a paradigm change as the emphasis shifts from collection sensors to a common operational picture based on seamless access to information. To accomplish this new paradigm, new technologies are required that simultaneously provide the Warfighter with "fly-by-wire" access to data and manage the finite system resources. The RIME architecture provides an early example of these next-generation systems. By allowing the Warfighter to explicitly quantify his information needs and the utility of that information together with Commander's policy defining the "rules" for allocating resources, the RIME system optimizes the resources to increase mission success.

New military doctrine based on information management versus network management together with additional research and development is required to make these systems a reality. Until the Warfighter is confident that mission-critical data is being delivered in a timely fashion, changes to the current bandwidth-centric management of assets is difficult. To increase users' confidence and develop stable high-speed information management systems, advances are needed in visualizing information flow, optimizing constrained resources, and efficiently processing large volumes of complex information requests.

Utility-Based Management of Information Dissemination[1]

Dr. Brian DeC.eene
TASC, Inc.
55 Walkers Brook Drive
Reading, MA 01867

Abstract

The exponential growth of the information content available to the user combined with the pervasive use of network-enabled light-weight personal computing devices (such as wearable computers and the commercial PalmPilot) requires technologies that maximize the utility of the information delivered. Unlike research areas such as advanced sensors, data-mining, and data-correlation which increase the availability and quality of information, information dissemination and management (IDM) addresses how user's information needs are delivered given end-host and network limitations. This is particularly important in dynamic mission-critical situations where real-time information must be delivered to the appropriate users in real-time despite the fact that the source of the information is not know a-priori and network resources are over subscribed.

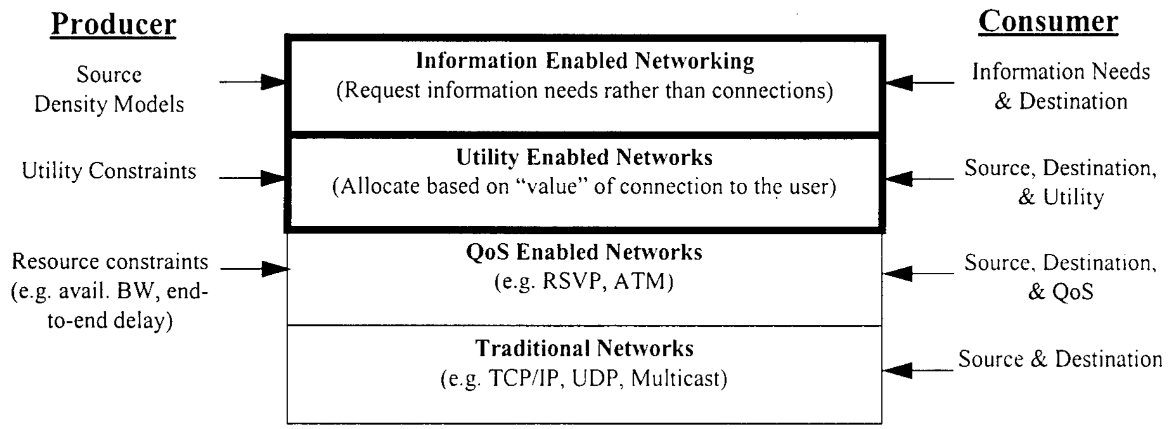
Recently, researchers have been investigating highly-scalable IDM architectures and the algorithms required to maximize the delivery utility of delivered data. This paper defines an information management architecture that has been developed by TASC and a supporting mathematical framework.

1. Introduction and Motivation

Achieving mission-critical situational awareness through information dominance requires timely and efficient delivery of information in response to changing conditions. Whereas improved sensor collection platforms and data processing techniques such as data mining enrich the depth and breadth of available information, they do little to ensure that the right information is delivered to the right users in a timely fashion to ensure overall mission success. Rather, most systems allocate and manage their resources on a first-come/first-serve basis with users competing for the available resources. As a result, information is distributed according to the time of the user's request rather than the importance of the information that is being conveyed. Although recent developments in Quality-of-Service (QoS) guaranteed networks help alleviate this problem, they are unable to consider how the connection is being used in determining which information is distributed to whom.

During a crisis, the problem is further aggravated by the observation that the necessary information not only exceeds the available capacity of the network resources but is often generated by unanticipated sources. Hence, static doctrine which dictate a specific chain of information dissemination can introduce costly delays and potentially misrepresent situations to the end user. By emphasizing the management of information over the management of connections, it is possible

[1] This work was partially funded by DARPA/ITO contract DABT63-99-C-0018.



Producer
Source Density Models
Utility Constraints
Resource constraints (e.g. avail. BW, end-to-end delay)
Information Enabled Networking
(Request information needs rather than connections)
Utility Enabled Networks
(Allocate based on "value" of connection to the user)
QoS Enabled Networks
(e.g. RSVP, ATM)
Traditional Networks
(e.g. TCP/IP, UDP, Multicast)
Consumer
Information Needs & Destination
Source, Destination, & Utility
Source, Destination, & QoS
Source & Destination

Figure 1: Utility-Based Information Service Stack

to allow the system to route information to the required users without a-priori knowledge of the source of the information.

To address the requirements of rapidly adapting to unanticipated events and globally maximizing the value of information delivered to the end-user for mission success, two additional technologies are required over the current state-of-the-art. These are utility and information management services as shown in Figure 1.

The introduction of "utility" on the existing QoS networks allows users to express the importance of their request to the overall success of the mission. Thus, rather than allocating QoS on a first-come/first-serve basis, the network allocates resources to maximize the total utility of the system subject to the physical limitations such as available bandwidth and latency. Another advantage of utility is that it allows users to quantify the characteristics of various alternatives to their ideal request. For example, while a bandwidth intensive connection may provide the maximum value to a particular user, utility allows the user to state that a connection at half the total bandwidth has some value to the user. Furthermore, this "what-if" capability inherent within utility allows utility-enabled networks to rapidly react to unanticipated demands on resources without entirely severing critical established connections.

Information-enabled networking is a shift in paradigm whereby consumers submit their information requirements rather than a specific source/destination pair. The network then defines channels that ensure that the user receives his required information, informs the users which channels to subscribe to, and directs the producers to put the approporiate information on each of the channels. Since these information needs are also expressed in terms of the utility of the information to the success of the mission, the network continues to dynamically adapt ensuring optimal informational awareness.

This paper mathematically formalizes the utility-based information management problem and explores some of its ramifications. This is based upon preliminary work performed by the AICE Functional Control Board[AICE-FACB99][Richardson99]. In Section 2. the notions of network connections with QoS allocations are defined and extended to define utility. Next, Section 3. defines utility constraints and their impact on resource management. Section 4. extends the previous definitions into the information domain and formalizes the additional services available at that layer.

2. Network Connections and Utility Functions

Define S and D to be the set of available sources and destinations respectively. We further define S* and D* as the set of all subsets of sources and destinations respectively[1]. Using this notation, we can define an arbitrary connection between a set of sources and their corresponding destinations as the ordered pair [S,D]. where S and D are non-empty. Similarly, a unicast connection ties a single source to a single destination. This is represented as [s,d] where s and d are elements of S and D. It follows that multicast connections ([s,D]) and comcast connections ([S,d]) can also be expressed.

Climbing the stack of Figure 1, the network community is actively investigating techniques for reserving network resources on a connection to ensure a particular quality of service. Examples of network resource reservation protocols include RSVP and ATM. Let us define Q as a set of QoS points that satisfy a user's connection requirements. Thus, a multicast session that has been assigned a specific QoS can be denoted by the triplet [s,D,q]. Alternately, a unicast connection that can tolerate any one of many QoS parameter sets is specified by the triplet [s,d,Q]. If Q* is the set of all subsets of Q, we will define the space of these ordered triplets (called connections in this paper) as $C^* \equiv S^* \times D^* \times Q^*$ such that any end-to-end QoS connection is an element of this set.

Note that in this formulation, we did not specify what constitutes an element of these sets. This is because it often depends upon the support provided by the underlying infrastructure. For example, under RSVP, users may specify their quality of service requirements in terms of rate and slack (end-to-end delay)[Wroclawski97]. Hence, the dimensions on Q would be these attributes.

While the ordered triplet fully defines a connection and its service characteristics, it fails to characterize the value of different alternatives to the user. For example, if the network is unable to support a particular multicast session as specified but would be able to support the request at either a reduced QoS or by eliminating some of the destinations, the single ordered triplet would be insufficient to enable the system to decide. For this reason, we define a utility function which weights the various connection alternatives.

Definition: The *channel utility function* for request x is denoted $u_x(C^*) \rightarrow \Re$ and satisfies the following properties:

- Maps a class of connections onto the real numbers. Notionally, this measures the utility of the worst-connection within the set.

[1] Throughout this paper, we let capital letters denote the set of all possible elements in a space whose components are denoted by the corresponding lowercase letter (e.g. $x \in X$). We further denote the abstract space of all subsets of X as X^*.

- All connections have a non-negative utility. I.e. $0 \le u_x(C)$ for all $C \in C^*$. Connections have zero utility if and only if they have no value to the user.
- Constraining a connection can only reduce the utility. I.e. if $A \subseteq B$, then $u(A) \le u(B)$.
- If two subconnections exactly recreate the original connection, then the total utility is the same as the original connection. I.e. if $A \cap B = \varnothing$, then $u(A \cup B) = u(A) + u(B)$.

As a mapping to the real numbers, utility does not directly imply the level of network loading induced by the connection but rather the usefulness of the connection if a particular QoS is provided. Thus a bandwidth intensive video application may give the same utility value as an audio connection but require 100 times more bandwidth. The network may decide to support 100 audio requests over a single video request so that the total system utility is higher.

Formally, the objective of the utility-enabled network is to optimize the total utility achieved subject to the physical infrastructure limitations. That is, given N requests for service, find $\begin{bmatrix} c_0 & c_1 & \dots & c_N \end{bmatrix}$ to maximize $\sum_{i=0}^{N} u_i(c_i)$. We shall define this as the *utility-based resource optimization problem.*

To illustrate the use of utility, consider the following example. Three users request connections where the utility is a linear function of the bandwidth provided. That is, decreasing the bandwidth decreases the utility until the connection is no longer useful. Providing bandwidth in excess of a particular value also provides no additional improvements in the utility of the data received[1]. The utility functions for these three users is illustrated in Figure 2. Optimizing these utilities subject to the constraint that the total consumed bandwidth must be less than a particular value yields the operating points shown in the graph on the right.

From this example, we can make a couple of observations. First, constant over a range of connections, then resource allocation should allocate the smallest value in that range. Increasing the bandwidth assigned to user 3 consumes additional network resources without increasing the utility. Along the same lines, if a utility is too restrictive, then it is less likely to receive any resources. For example, if user 1 had demanded precisely six units of bandwidth, then no resources would have been allocated to that request in the previous example. However, since the user identified that there was partial value to having less bandwidth, a connection was provided with three units of bandwidth. Considering the plot from a different angle, the utility function graceful degrades performance as the bandwidth decreases although interrupted connections may be reestablished.

Another observation from this example is that a utility function that is non-zero at only a single point (i.e. a traditional QoS request) is likely to fail. If user 1 had requested a connection of five bandwidth units with utility of four and zero otherwise, he would not have received any allocation

[1] This example could have also been done using some of the other attributes of a connection. For example, dropping destinations from the requested connection could result in reduced utility until it no longer made sense to establish the connection. One can think of this as having a meeting where attendance of some members is desired but not required.

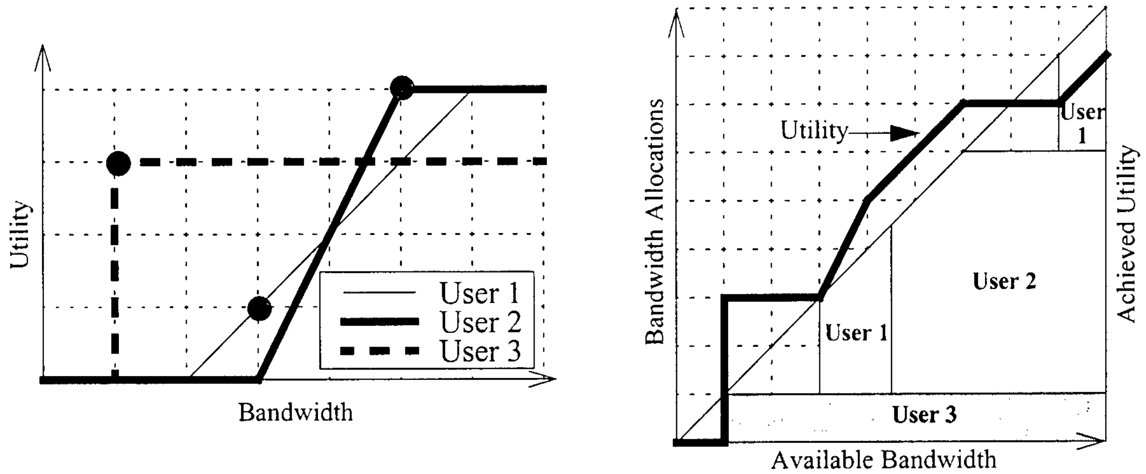


Figure 2: Utility-based Optimization.
Given the three user's utility functions on the left, the optimal allocations as a function of available bandwidth is shown on the right. The bold line overlaid shows the utility achieved under the specified bandwidth allocations. For example, when nine units of bandwidth are available, then the user 1 is allocated 3 units; user 2 is allocated 5 units; and user 3 is allocated 1 unit. The achieved utility is 8. These operating points are shown on the left diagram.

in this example. Along the same lines, a request that is constant (see user 3) will result in the minimum allocations at all times and is therefore equivalent to a single point request. Naturally, users could game the system by choosing utility values higher than their counterparts. Managing this is the topic of the next section.

3. Utility Constraints / Policy Functions

In utility-based resource management, requests can be preempted by increasing the utility sufficiently high to ensure that all other requests have no substantial value since there is no cost or limits placed on the utility function defined. The solution is to introduce constraints on the utility functions.

A utility constraint (or policy) places a bound or cost on an individual's utility function. Upon admission, a channel request is compared against these constraints and may be rejected or modified. Furthermore, a change in a utility constraint may cause an existing submission to be modified and change the operating point generated through the utility-based resource optimization discussed previously. Before formalizing the definition, let us consider some examples of policies to develop intuition.

Example: Thresholding
The simplest form of policy is to restrict any user from exceeding a particular maximum utility. Thus, whereever the utility function exceeds a specified threshold, it is mapped to the maximum allowed value. If in the prior example, user 2's utility function of $u(\mathrm{BW}) = 2\mathrm{BW} - 6$ had not been truncated to four, no bandwidth would have been allocated to users 1 or 3. We will denote this function as $T(u, a)$ which truncates the utility function to values below a.

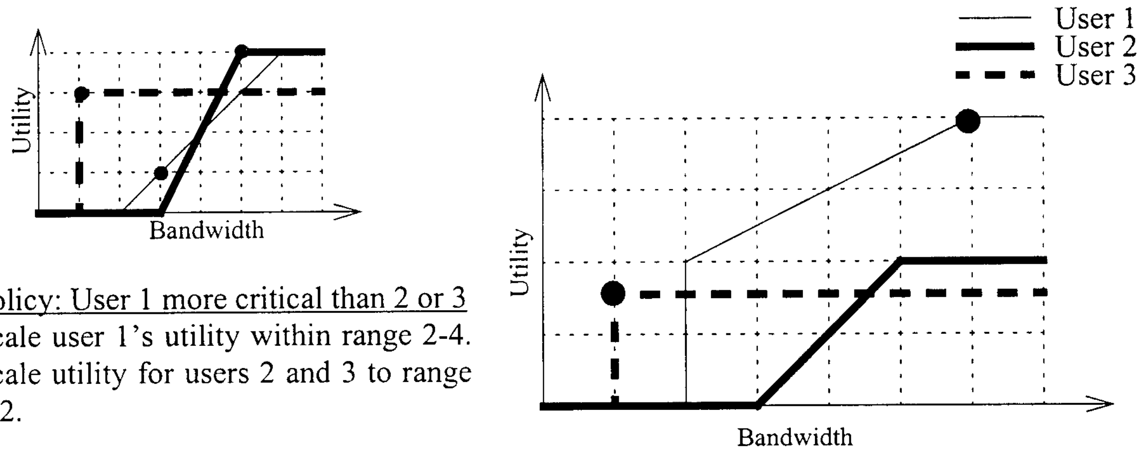
Example: Scaling (or Ranking)
The difficulty with thresholding is that it fails to prioritize one request over another without destroying the original shape of the utility function. Fortunately, scaling enables certain requests to receive preferential treatment over other requests without losing the shape of the utility function. For example, requests to support a life-threatening activity should take precedence over normal administrative activities. While both tasks can make requests for connections, policy functions ensure that the appropriate resources are allocated to the critical task first. An example is illustrated in Figure 3. In this case, since user 1's mission is more critical than user 2 and user 3, the policy weights user 1's utility over the others. Therefore, the desired behavior is that user 1's utility function is mapped into the range [2-4] while users 2 and 3 are mapped into the range [0-2].

Definition: Let $u_{\min}$ and $u_{\max}$ be the minimum and maximum values of the utility function u. Then, the *scaling function* into the interval [a,b] is given by

$$R(a, b) \bullet u(C) = R(u, a, b) + a = (b-a)\left[\frac{u(C) - u_{\min}}{u_{\max} - u_{\min}}\right] + a \qquad (1)$$

Thus, using $R(u_1, 2, 4)$ for user one; $R(u_2, 0, 2)$ for user two; and $R(u_3, 0, 2)$ for user three, the resulting utilities are shown on the right of Figure 3. After utility-base resource optimization, user 1 is allocated 5 units (versus 3 in the previous example); user 2 is allocated no bandwidth (versus 5 units previously); user 3 has no change; and three units are unused.

Note that, while policies modify the utility function, they are not constrained by the physical limitations of the underlying infrastructure. For example, in the illustration above, the utility was determined independent of the fact that only nine units of BW were available in the system. These considerations are considered additional constraints on the optimization problem; not part of the



Policy: User 1 more critical than 2 or 3
Scale user 1's utility within range 2-4.
Scale utility for users 2 and 3 to range 0-2.

Figure 3: Policy Applied to Channel Utility
Given the previous users utility functions, the policy causes user 1 to be weighted higher than the other users. Consequently, the allocation of resources shifts to emphasize user 1. The resolved requests and new operating points are shown on the right.

policy formulation.

Example: Integral Scaling
An alternative to scaling is to use a credit-based system where each request distributes a fixed amount of utility over the desired QoS space. Thus, specifying a certain QoS with a high utility reduces the utility available for the other potential QoS values. We will define this function as

$$I(u) = u(C)/\left(\int_{C^*} u(C)\right) \tag{2}$$

From these example, we shall now generalize the definition of a policy function and explore its properties.

Definition: The i-th *channel policy function* that applies to user x maps a utility function into another utility function (denoted $\alpha_i(U^*) \rightarrow U^*$) and satisfies the following properties:

- Policy-corrected utilities are compliant with the policy. That is, $\alpha_1 \bullet \alpha_1 \bullet u_x = \alpha_1 \bullet u_x$
- Policies are associative. That is, $\alpha_1 \bullet (\alpha_2 \bullet \alpha_3) \bullet u_x = (\alpha_1 \bullet \alpha_2) \bullet \alpha_3 \bullet u_x$.
- There exists an identify policy that does not change the utility function and a zero policy that assigns the utility function to zero everywhere (i.e. assigning resources has no value).

Theorem : Truncation, scaling, and integral-scaling functions are all valid policy functions.

Proof. We will show this for the scaling function only and leave it as an exercise for the other functions. The key property to show is the first item. It is easy to note that upon applying the transform once, the minimum and maximum of R(u,a,b) is a and b respectively. Thus

$$R(a, b) \bullet R(a, b) \bullet u = (b-a)\left[\frac{R(a, b) \bullet u - a}{b-a}\right] + a = R(a, b) \bullet u \ . \tag{3}$$

In defining the utility-constraint function we were only interested in policies that *applied to user x*. However, we did not identify how policies functions and connections are associated; particularly given that neither the policies functions nor connection requests are known a-priori. Consequently, we need to define a technique for correlating policies against requests. This is accomplished by attaching a "key" to both the utility function and each channel policy function such that if the request's key matches the policy, then the policy is considered applicable. Specifically, let us define the binding space, B, as the set of keys that are associated with each request and each policy function.

Definition: A *policy-constrained channel request* (or simply *channel request*) is defined as a utility function specifying a set of possible connections and a binding space that associates policies to the request. This is denoted $\xi_x = [u_x, B_x]$.

Definition: A channel policy is defined as a channel request policy function together with a binding space ($p_i^y = [\alpha_i, B_y]$) such that p_i^y applies to ξ_x if and only if $B_x \cap B_y \neq \varnothing$.

To illustrate, consider a system administrator who wishes to place policies on three destinations such that their access to a pair of database servers is managed. Specifically, server 1 primary function is to serve destination 1. All requests by that destination should get precedence over requests from the other destinations. However, the other destinations are permitted access to server 1 if it is not in use by destination 1. On the other hand, server 2 is intended to support destinations 2 and 3. Furthermore, the total utility for any request from either destination 2 or destination 3 must be less than one to ensure that the destinations compete fairly.

In this case, the binding space is defined as the source/destination space ($B^* = S^* \times D^*$). The policies are

1. $[R(u, 1, 2), \{(s_1, d_1)\}]$: Scale utility between 1 and 2 for connections between server 1 and destination 1.
2. $[R(u, 0, 1), \{(s_1, d_2), (s_1, d_3)\}]$: Scale utility between 0 and 1 for connections between server 1 and destinations 2 or 3.
3. $[0, \{(s_2, d_1)\}]$: Assign no utility to connections between server 2 and destination 1.
4. $[I(u), \{(s_2, d_2), (s_2, d_3)\}]$: Use finite credit for requests to server 2.

Although this example constructed the binding space from the previously defined source and destination spaces, this is not always the case. For example, [BADD98] applied policy according to the user's command hierarchy and mission alone. In this case, every request for services included an identification of the user causing users to be broken down into hierarchical sets -- all users within the same unit are one set; all users within units within a particular battalion defines another set; and so forth. With each policy specifying a set of users through units, battalions, etc., the binding between the request and the constraint could be defined.

Yet, the binding alone is insufficient to apply policy to a particular request since utility constraints do not generally commute. (i.e. $\alpha_1 \bullet \alpha_2 \bullet u_x \neq \alpha_2 \bullet \alpha_1 \bullet u_x$). For example, it is easy to see that $T(1) \bullet R(0, 2) \bullet u \neq R(0, 2) \bullet T(1) \bullet u$ for user 1's utility of Figure 2. As a result, it is necessary to define an ordering function ($\Gamma(i)$) that specifies the order that the policies are applied to a user's request. For example, military command structures are typically hierarchical -- each individual is subject to the policies of his commander. In this case, the ordering function is defined by the position in the command structure of the person submitting the policy. Naturally, this rule does not help resolve multiple policies submitted by the same commander. In this case, additional rules are required to determine the ordering.

Using the ordering function, the policy-resolved request for user x with M applicable policies is given by

$$\alpha^x_{\Gamma(0)} \bullet \alpha^x_{\Gamma(1)} \bullet \alpha^x_{\Gamma(2)} \bullet \ldots \bullet \alpha^x_{\Gamma(M)} \bullet u_x . \tag{5}$$

Clearly, if we define a resolved policy as $\tilde{\alpha}^x_m \equiv \alpha^x_{\Gamma(0)} \bullet \alpha^x_{\Gamma(1)} \bullet \ldots \bullet \alpha^x_{\Gamma(m)}$, then it is easy to show from the properties above that

$$\alpha^x_{\Gamma(0)} \bullet \alpha^x_{\Gamma(1)} \bullet \ldots \bullet \alpha^x_{\Gamma(M)} \bullet u_x = \tilde{\alpha}^x_1 \bullet \tilde{\alpha}^x_2 \bullet \ldots \bullet \tilde{\alpha}^x_M \bullet u_x = \tilde{\alpha}^x_M \bullet u_x \tag{6}$$

In other words, replacing the originally submitted policies with their resolved policy equivalent does not change the behavior of the system. For this reason, pre-calculating various resolved policies are being actively considered as a technique for reducing the processing time required to resolve a request against policy.

4. Information-enabled Networking

A limitation of the previous formulation is that it does not exploit redundancies in the requests for information. Different users who independently request connections for the same information cause the system to send the data individually to each user and waste network resources. This limitation can be overcome by a simple extension of the previous binding space to provide content-based processing services.

Information (or content-based) processing extends the traditional channel request/management services by exploiting details about the information that is intended to flow over the connection. Some of the services that are available under content-based processing include:

- Information-based Utility and Policies: Rather than defining the utility of various connections, utility is defined based on the value of the information. Along the same lines, information resource policies shape the information utility function and manage the desired system behavior. For example, a requestor may specify that data corresponding to a particularly important geographic location is mission-critical over other information requests.
- Automatic source selection: Since consumer applications express their information interests and producer application summarize their available information through density models, the system can dynamically identify which sources should be feeding the constructed channels. This eliminates the need for a-priori knowledge of the available sources.
- Source-based Utility Modification: Under the previous model, the requestor is required to define the utility function based on some a-priori information about the characteristics of the information that will be sent. In the information domain, the network can determine the necessary QoS utility function by combining the source density models with a user defined utility in terms of the information itself.
- Dynamic Channelization: Because the content of the connection is known, the system can improve resource usage by aggregating similar requests onto a shared channel.
- Tailored control of sources and destinations: Unlike broadcast models where users passively subscribe to broadcasts containing specific content (e.g. news channel CNN versus sports

channel SPAN), content-based processing allows the system to precisely control the content on the channels such that only relevant information is transmitted. Thus, if a user is only interested in news reports about a particular event, then only those reports would be transmitted versus CNN that broadcasts all news events even if no one is interested.

We will explore each of these services in detail in the following sections.

4.1 Information-based Utility, Resource Policy, and Access Policy

We begin by extending the notions presented in Section 2. and Section 3. by defining the exchange characterization space which encompasses both the binding space and the connection space($E^* \supseteq B^*$). Furthermore, this space includes any additional parameters that enable a user to describe their information needs.

Example: An local weather forecasting application requires recent weather measurements for the immediate surrounding region. If measurements such as barometric pressure and temperature are provided within 15 minutes of capture, then the information is very useful. Information delivered within an hour of the measurement has limited usefulness. Information delivered within an hour for regions beyond the local area also has limited usefulness. After an hour, all information is too stale to be used in the application.

As a result, the user submits an information request with a utility of

- Weather data within 100 miles of station delivered within 15 minutes has utility of 10
- Weather data within 100 miles of station delivered within 1 hours has utility of 5
- Weather data within 1000 miles of station delivered within 1 hours has utility 5

Rather than specifying specific connections to various sites or identifying the connection's required QoS, the applications needs are expressed abstractly allowing the system to solve for the additional required data. In this case, the request does not need to know a-priori the bandwidth required to support this request. Furthermore, if multiple applications are interested in the same weather data, they can all be served with a single multicast channel saving network resources.

From example, it is easy to see how the previous definitions of the previous section are extended to support information management. Specifically, these are shown in Table 1. As can be shown,

Table 1: Information-Enabled Basic Definitions

Information Domain	Definition	Analogy
E^*	Exchange characterization. Also called the Information Space Region (ISR) or Metadata Space Region (MSR).	B^*
$v(E^*) \rightarrow \Re$	Information utility function. Defines the "value" of information to the user.	$u(B^*) \rightarrow \Re$
$\Xi_x \equiv [v_x(E^*), E_x]$	Policy-constrained information request (or simply information request).	$\xi_x \equiv [u_x(B^*), B_x]$

Table 1: Information-Enabled Basic Definitions

Information Domain	Definition	Analogy
$\alpha(V^*) \rightarrow V^*$	Information policy function.	$\alpha(V^*) \rightarrow V^*$
$p_x \equiv [\alpha_x(V^*), E_x]$	Information resource request policy.	$p_x \equiv [\alpha_x(V^*), E_x]$

these definitions adhere to the properties outlined earlier with the appropriate change in variables.

4.1.1 Information Utility

The information utility function places value on various information independent of how that information is delivered. Like the previous utility definition, this is subject to the same properties as below except that the connection space is replaced by the exchange characterization space.

4.1.2 Information Policy

This capability follows naturally from the extension of the binding space into characterization space. Its advantage is that the connection is now tied to how it will be used as well as who/what is requesting the connection. As before, this adheres to the properties of policy outlined previously.

4.1.3 Access Policy

Information that has no value to a particular user is assigned a utility of zero. In the same fashion, an information policy can modify a user's utility function such that certain information is given zero utility. The effect is that, during resource allocation, it is unlikely that any zero utility information will be allocated to a channel. However, as we shall see in Section 4.3, channelization may transmit information that has no utility to a particular user on a shared channel because, by sharing the channel, the overall utility of the system is improved. *Thus, a utility of zero does not guarantee that the information is not sent to the user.*

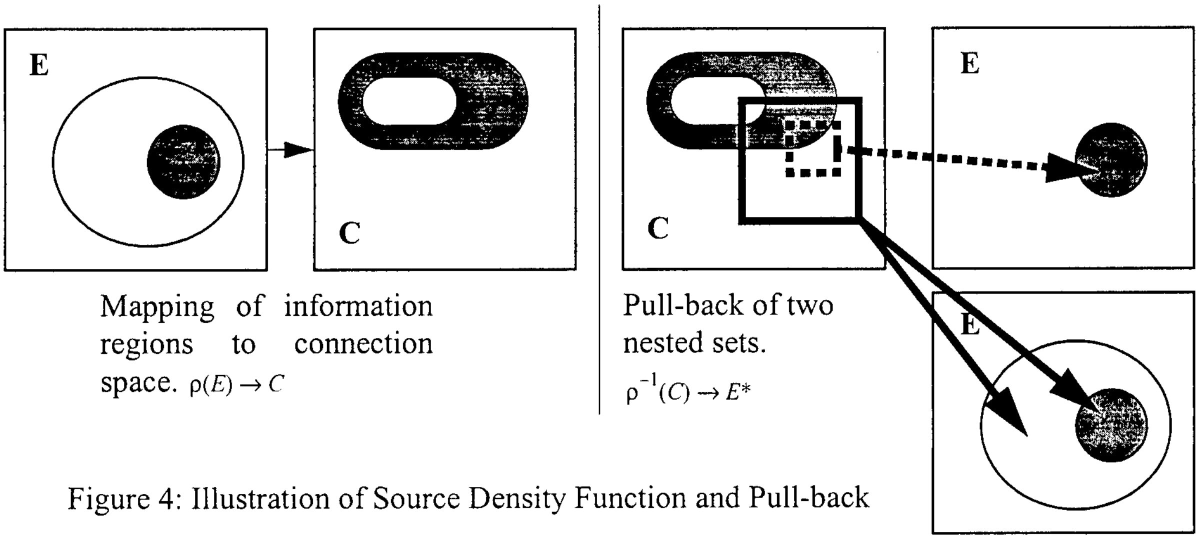
An access policy differs from an information policy in that it expressly prohibits information from being delivered to a particular user. Thus, access policies constrain the available channelization solutions as is discussed later.

4.2 Automatic Source Selection and Channel Utility Construction

Definition: The *source density function*, $\rho_a(E) \rightarrow Q$, for source "a" defines the QoS required to deliver information E. This is subject to the following properties

- $\rho_a(E) = \varnothing$ if and only if source "a" contains no information satisfying E
- If $E_1 \subseteq E_2$ then $\rho_a(E_1) \supseteq \rho_a(E_2)$.

The second bullet above is a consequence of the observation that increasing the requested infor-



Mapping of information regions to connection space. $\rho(E) \to C$

Pull-back of two nested sets. $\rho^{-1}(C) \to E^*$

Figure 4: Illustration of Source Density Function and Pull-back

mation requires more resources and therefore decreases the possible set of QoS values that are sufficient to guarantee delivery. In other words, any QoS that is sufficient to send E_2 is sufficient to send E_1 .This is illustrated in Figure 4.

From this definition, we define the pull-back of QoS as the information that can be delivered given a particular QoS as the set of sets whose image under the density function is Q (i.e. $\rho_a^{-1}(Q) \to E^*$). Note that this returns a set of exchange characterizations that are all possible to satisfy by the particular group of QoS parameters. Let us define $\hat{u}(Q) \equiv \max v(E) : E \in \rho_a^{-1}(Q)$.

Theorem : $\ddot{u}(Q)$ is a channel utility function.

Proof. By definition, $\ddot{u}(Q) \geq 0$ since $v(E) \geq 0$. Consider Figure 4. If $A \subseteq B \subseteq C^*$, then it follows that $(E^*_A = \rho_a^{-1}(A)) \subseteq (E^*_B = \rho_a^{-1}(B))$ -- any element that is in the pull-back of A is also in the pull-back of B -- so $\ddot{u}(A) \leq \ddot{u}(B)$. Finally, let A and B be elements of connection space with no intersection. Then let $\tilde{\rho}_a^{-1}(A)$ and $\tilde{\rho}_a^{-1}(B)$ be the subset of the pull-backs of A and B that have the maximum utility of all pullbacks. Clearly these sets cannot intersect. Then we have

$$\hat{u}(A \cup B) = v(\tilde{\rho}_a^{-1}(A) \cup \tilde{\rho}_a^{-1}(B)) = v(\tilde{\rho}_a^{-1}(A)) + v(\tilde{\rho}_a^{-1}(B)) = \hat{u}(A) + \hat{u}(B) .$$

Hence, we can construct a request's channel utility function without having to know a-priori the service required to deliver the data!

4.3 Channelization

Given that the system knows not only the utility of the information but also the content to be transmitted over the connection, channelization looks for opportunities to aggregate users onto shared channels reducing redundant transmissions. The result of this analysis consists of a set of channel definitions along with a subscription rule that defines which channels a user needs to join in order to satisfy their information needs.

Definition: A *channelization algorithm* ($\chi(\Xi^*) \rightarrow \Xi^*$) maps a set of information requests into another set of information requests such that

- The union of all the information covered by the new sets contains all information contained in the original sets. That is $\bigcup_{\Xi \in \Xi^*} E_{\Xi} \subseteq \bigcup_{\Xi \in \chi(\Xi^*)} E_{\Xi}$. We say that the constructed channels *cover* the previous information requests. We further say that a covering is *orthogonal* if the intersection between any two different sets after channelization is null. A covering is *exact* if the equality above holds.
- Aggregation increases the utility. That is, for every request, $x \in \Xi^*$, there exists a corresponding channel, $y \in \chi(\Xi^*)$, such that $v_y(E_x \cap E_y) \geq v_x(E_x \cap E_y)$.

Observe that the second property does not require that all channels that intersect the requests information region has to have a higher utility -- only that at least one channel exists that does. For the remainder of this paper, we shall combine utility functions by adding them together[1].

Definition: The *subscription rule* for a given channelization algorithm defines which channels a request needs to subscribe to in order to fully satisfy their information needs. This rule is *precise* if the union of the exchange characteristics for the subscribed channels equals the original exchange characteristic. The *standard subscription rule* is that each request subscribes to all channels that intersect the requests exchange characteristic.

Lemma : The standard subscription rule is sufficient for any orthogonal channelization algorithm.

The interpretation of orthogonal, exact, and precise bears additional discussion. If a covering is orthogonal, then no information is common on two different channels. As a result, all information is broadcast exactly once. On the other hand, exactness implies that only information that at least one user has requested is sent. Thus, evening news broadcasts are neither orthogonal (since different stations carry the same stories) nor exact (since the broadcast is independent of whether anyone is listening). Preciseness measures the amount of excess information that a user may receive when subscribed to a particular channel. For example, a user that is only interested in tomorrows weather and subscribes to the evening news is not precise since they will also receive other new stories.

Let us briefly examine some simple channelization algorithms illustrated in Figure 5.

[1] The combination of utility functions is important in that it effects the “fairness” of the competition between various channels. For example, are 100 items with utility 1 aggregated together more important than a single very important request at utility 99. This issue is analogous to the issue of TCP/IP fairness within the multicast community.

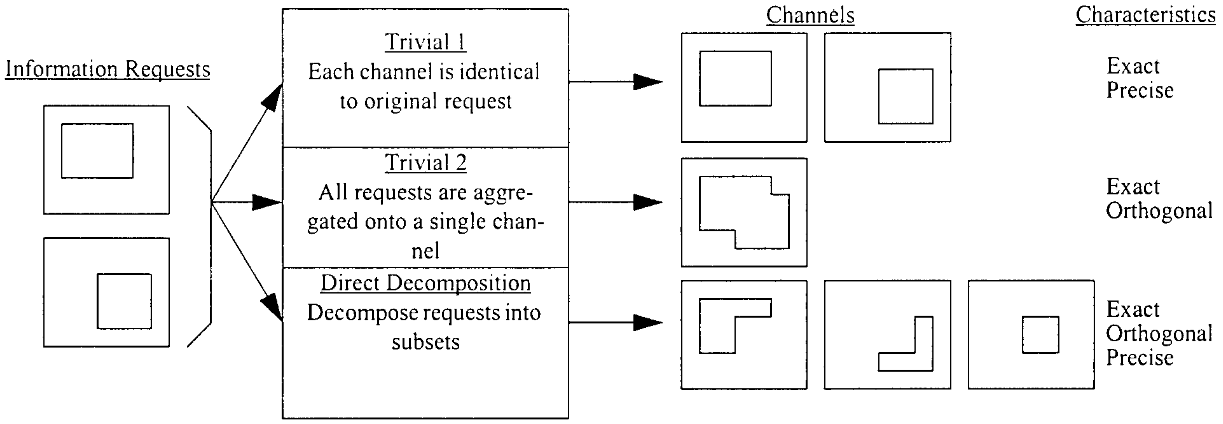


Figure 5: Sample Channelization Algorithms

4.3.1 Trivial Channelization Algorithms (No Aggregation and Full Aggregation)

The trivial channelization algorithm maps each request onto itself ($\chi_{\text{trivial1}}(\Xi_1, \Xi_2) = \{\Xi_1, \Xi_2\}$)

with a subscription rule that each request joins its corresponding channel. Clearly this algorithm is exact and precise but not orthogonal as overlapping requests for data will result in the same data being sent twice.

Another trivial channelization algorithm maps each request onto a single channel: $\chi_{\text{trivial2}}(\Xi_1, \Xi_2) = [u_1 + u_2, E_1 \cup E_2]$. This algorithm is both orthogonal and exact. However, it is not precise since subscribing to the common channel results in excess information being received.

4.3.2 Direct Decomposition

Neither of the previous two algorithms took into consideration the interrelationships between the various information requests. This situation was investigated by [Zabelle98] and the results are extended here under the current framework.

Given two utility-based requested for information, we define a channelization operator that generates three disparate channels given two channel requests. Specifically,

$$\chi_{\text{opt}}(\Xi_1, \Xi_2) \equiv \{[u_1, E_2^C \cap E_1], [u_2, E_2 \cap E_1^C], [u_2 + u_1, E_2 \cap E_1]\} \qquad (7)$$

Channelization of more than two requests is defined by recursively applying the above equation. The standard subscription rule is used. It follows from this definition that optimal channelization

- Satisfies the properties of the channelization algorithm definition.
- This algorithm is exact, orthogonal, and precise.

- Commutative (i.e. $\chi_{\text{opt}}(\Xi_1, \Xi_2) = \chi_{\text{opt}}(\Xi_2, \Xi_1)$)
- Associative (i.e. $\chi_{\text{opt}}(\Xi_1, \chi_{\text{opt}}(\Xi_2, \Xi_3)) = \chi_{\text{opt}}(\chi_{\text{opt}}(\Xi_1, \Xi_2), \Xi_3)$)
- Combining two requests with non-overlapping information needs does not change the utility.

  (i.e. If $E_1 \cap E_2 = \{\varnothing\}$ then $\chi_{\text{opt}}(\Xi_1, \Xi_2) = \{\Xi_1, \Xi_2, \varnothing\}$).

Theorem : The number of channels in any orthogonal and precise channelization algorithm must be equal to or greater than the number of channels in the direct decomposition.

Proof. Assume the contrary. Then there must be at least one channel (call it X) that intersects at least two direct decomposition channels. However, for any two direct decomposition channels there is at least one requestor that is interested in the data in one set of the direct decomposition but not interested in the other set. Due to orthogonality, the standard subscription rule is sufficient and this request would subscribe to X and receive information that was not requested. Therefore, the algorithm is not precise -- contradiction.

Lemma : The direct decomposition is the best orthogonal, precise channelization algorithm.

Unfortunately, while the direct decomposition is theoretically optimal, the number of channels grows exponentially as 2^N where N is the number of requests. Therefore, non-optimal solutions are required. The area of constrained channelization is an active research area.

4.4 Tailored Source/Destination Control

Density model defines what information is available on each source and the required QoS necessary to satisfy it. Channelization establishes what content should be broadcast over each individual channel. Therefore, information management select sources and specifies what information they should send.

5. Comparison against Existing Systems

Numerous utility-based and information-based network architectures have been proposed in recent years as a natural evolution from the current systems. While not refined formally to the knowledge of this author, they have demonstrated the potential of these services and warrant discussion.

The DARPA/ISO Battlefield Awareness and Data Dissemination (BADD) program developed and demonstrated a functional and software architecture demonstrating many of the information management services. Specifically, the BADD system enabled users to express information requests with constant utility value (precedence). Access policy was applied to these requests and the precedence value potentially cropped by a single resource policy that specified the maximum allowed value for a particular mission. The validated profile was then channelized using the non-aggregating trivial algorithm described earlier[1]. Information types such as file, WEB, streaming video, and database data were all managed by this system.

BADD was a success because it proved the feasibility of information management services and developed a set of software requirements that capture the characteristics of the functional architecture. However, the currently implemented algorithms for channelization and support for resource policies are limited in both performance and scalability. In addition, the BADD architecture lacked an underlying network infrastructure that would support utility-based resource reservation. Currently, the DARPA/ITO Agile Information Control Environment (AICE) program is addressing this technology gap to better manage the network resources through scalable utility-based services and supporting high-speed algorithms.

In addition, numerous companies have implemented some of these services -- particularly within the Web environment -- in direct response for the need to deliver more services to their customers using a limited networking resource. For example, TIBCO has developed an architecture with three layers: Informaton Dissemination & Event Nofication; Data Integration, Routing & Transformation; and Messaging [Tibco99]. These layers provide some of the information management capabilites with neither policy, channelization, nor utiliy. Other vendors, such as Netscape, have developed servers to proactively construct custom WEB pages based upon a user constructed profile.

6. Conclusion

While projects such as the BC2A and BADD have demonstrated the technical feasibility of IDM services, they have only begun to address the research issues necessary to achieve the full potential of IDM.

Bibliography

[AICEFACB99] AICE Functional Architecture Control Board, "AICE Functional Architecutre Overview," June 1999.

[BADD98] "Battlefield Awareness and Data Dissemination Phase IIC Objective Functional Requirements for Data Dissemination", Ver. 1.0c, Aug. 1998.

[Richardson99] J. Richardson, "A Cut at Refining Harry's 'Summary Document'", May 1999.

[Tibco99] TIBCO Product Release Information, www.tibco.com/products/active_enterprise/index.html. 1999.

[Wroclawski97] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," IETF RFC 2210, Sept. 1997.

[Zabele98] S. Zabele, "Channelization Techniques for Data Dissemination," TASC Working Document, TASC Inc., 55 Walkers Brook Dr., Reading MA 01867, April 1998.

[1] Some preliminary work in a containment-based channelization algorithm has been performed under BADD based on qualitative assumptions.

EXAMINING THE CONTAINMENT ALGORITHM

Introduction. Consider the following very simple, abstract, model of an information dissemination system: There are N distinct information regions, and there are M users who may want some of those regions. Let A be an $M \times N$ matrix, called the user request matrix, whose rows represent users and whose columns represent information regions. If user m wants information region n, then the mnth element of A is 1, while if user m does not want information region n, then the mnth element of A is 0.

The problem is how to efficiently package the information for dissemination. That is, the problem is to combine information regions into channels which are then singlecast or multicast to various users. That packaging is described by two binary matrices. One, denoted by C, is called the channelization matrix. It is much like the user subscription matrix, except that it has one row per information channel, and that row has a 1 in each column for which an information region is carried by the channel in question. Matrix C is $K \times N$, where K is the number of channels. The other matrix is called the user subscription matrix, denoted by S. It is a binary $M \times K$ matrix with a 1 in row m and column k if user m subscribes to (receives) channel k. If each user is to get everything it requested, then there must be a 1 in each position of the product SC where there is a 1 in A.

One group of channelization methods constructs a single channel for each user. One extreme method in this group is to make one giant channel containing all the information regions wanted by every user, and broadcast that same channel to every user. This method may overwhelm some or all of the users with unwanted information, but it minimizes the total information sent. Essentially, this just places the problem of filtering information for individual users into the hands of the users themselves, who may lack the resources to accomplish it. A second extreme method of this group is to do all the filtering at the source, make a separate channel for each user, a channel containing precisely the information the individual user wants, and singlecast each of those channels to their respective users. This method may overwhelm the information system by sending many information regions repeatedly, but it eliminates the unwanted information received by the users.

A third extreme method, not in the group that sends only one channel to each user, is to create one channel for each wanted information region, and singlecast or multicast, as appropriate, those channels to the users requesting them. This method minimizes both the unwanted information and the total information sent, but it may require the management of very many channels, and the complexities of addressing those channels to differing subsets of users. If the ability of the information system to manage channels limits their number, either of the latter two methods may require more channels than the system can accomodate.

A way is sought that channels the information into a limited number of channels that, in some compromise sense, minimizes the unwanted information received by the users and minimizes the total information sent to all users. The way the information is channeled is often called an algorithm, referring to the computations needed to decide which information regions go into which channels and which users get which channels.

Containment. One possible compromise method—really no compromise at all— is to examine the rows of the user request matrix to find users with identical requests. A separate channel is then constructed for each unique request and is either singlecast to its only requestor or is multicast to its several requestors. Of course, any user that wants no information regions at all—any all-zero row of the user request matrix—may be disregarded. Such a situation would never happen in practice, for a user request would not be initiated until some information region was requested, but it may well happen in the methods for examining the packaging algorithms here. When the number of information regions is large, there may be very few or no identical requests, and this approach may accomplish little or nothing.

A further compromise—a real compromise—is to examine the rows of the user request matrix for rows that "contain" other rows. If all the requested information regions in row i are also requested in row j, then row j is said to contain row i. Another way to think about this is that if any element of row i is 1, while the corresponding element of row j is 0, then row j does not contain row i. Two rows that are equal will contain each other by this definition. In this compromise, termed the *containment algorithm*, contained rows of the matrix are so marked, except that one row of each set of equal rows will survive as uncontained. (An all-zero row will be contained by any other row, so this special case may usually be disregarded, a possible exception being when all the rows are all zeros. This is admittedly a very uninteresting nuisance case, but it can skew statistics when examining situations with very small numbers of users and information regions. When it happens, the containment algorithm declares all the rows to be contained.) A separate channel is then constructed for each uncontained user row and is sent to its user and to each other user contained by this one. This method is in the group of methods that sends only one channel to each user. When the number of information regions is large, there may be few or no user requests that contain those of other users, but it is not clear at the outset to what extent this may happen.

The question is: What savings in information sending is likely with the containment algorithm of packaging? The answer to this question obviously hinges on the user information requests. In some situations, significant subsets of users may have very similar requests.

Deterministic Considerations. One aspect of the question is capable of a deterministic answer. Given N information regions, how many uncontained user requests are possible? The answer is

$$N_{um}(N) = \binom{N}{N/2} \qquad (1)$$

In this formula, the binomial coefficient is indicated, and the lower number, $N/2$, is an integer. If N is odd, it makes no difference whether $N/2$ is rounded up or down, the result is the same.

The derivation of this result is straightforward. Consider a user request as a binary number having N bits, which are the elements of the user request row. For example, consider the simple case where $N = 3$. There are only 2^N possible distinct user requests (including the null request), 8 in this example, numbered from 0 to $(2^N - 1)$. These requests may be used to number the vertices of an N-dimensional hypercube, with the example case being illustrated in Figure 1. Here, each vertex contains any vertex below it on a shared edge, and is contained by any vertex above it on a shared edge. Containment "chains" are any upward paths along edges of the cube. In the example, vertex 5 (binary 101) contains vertices 1 (binary 001) and 4 (binary 100). That is, the cube vertices are grouped into $N + 1$ levels (here 4), with the level indicating the number of 1s in the binary representation of the vertex number. There are obviously

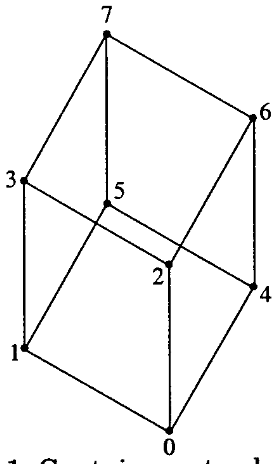


Figure 1: Containment cube $(N = 3)$

$$N_k = \binom{N}{k} = \frac{N!}{(N-k)!k!} \qquad (2)$$

vertices on the kth level, since there are this many ways to place k 1s into N bits. It is clear that none of the vertices at the kth level contain each other, and some thought will show that there can never be more uncontained user requests than there are vertices at that level that has the most vertices, leading to the formula above for the maximum number of uncontained user requests.

Perhaps more useful is the ratio of the maximum number of uncontained user requests to the maximum number of possible user requests, which is

$$r_{um}(N) = \frac{\binom{N}{N/2}}{2^N} \qquad (3)$$

This ratio tends slowly downward as N increases, although, when N is odd, the ratio is equal to that for the next higher even value. When N is 10, the ratio is about 0.25, while when N is 60, the ratio has fallen to about 0.1, as Figure 2 shows. This might look attractive, until it is realized that there are more than 10^{17} possible uncontained user requests for $N = 60$. The hope is that the ratio might indicate a potential for consolidating user requests by containment when there are only a tiny fraction of those possible.

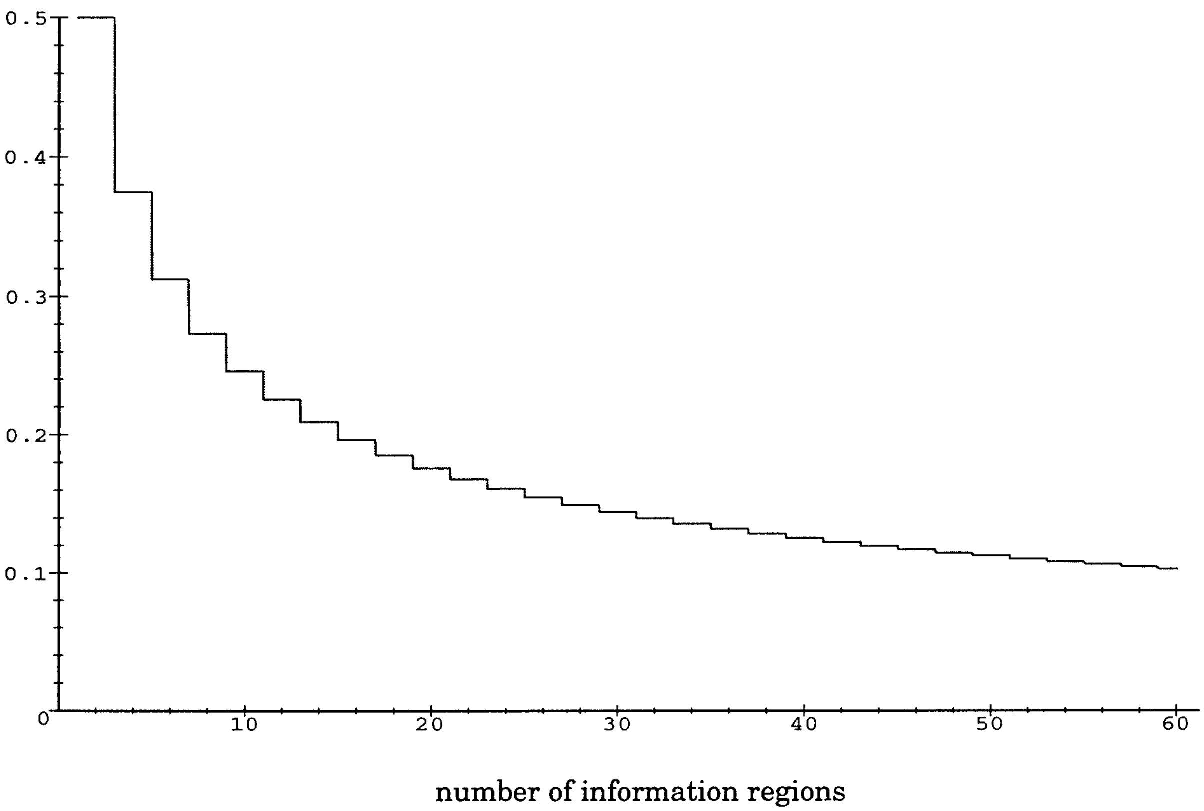


Figure 2: Ratio of maximum number of uncontained requests to the number of possible requests

A Probabilistic Formula Approach. Additional insight into the question of the potential utility of the containment algorithm might be obtained by considering random information requests. That is, suppose each element—0 or 1—of the user request matrix were chosen independently, at random, with a probability p of being 1, and a corresponding probability $q = 1 - p$ of being 0. Given that probability p, the number of users M, and the number of information regions N, how many uncontained rows of the matrix are likely to survive, and how many information regions will they likely have?

Elementary probability formulas do little to answer these questions. Define the probability P_{er} to be the probability that any two distinct selected rows of the user request matrix, say row i and row j, are equal through r elements. If two rows are equal through r elements, they must be equal through $r - 1$ elements, and the independently chosen rth elements must also be equal, so

$$P_{er} = P_{er-1}(q^2+p^2) = P_{er-1}(1-2qp) \tag{4}$$

$$P_{er} = (q^2+p^2)^r = (1-2qp)^r \tag{5}$$

The probability that two rows are fully equal (through all N elements) is P_e, where

$$P_e = P_{eN} = (1-2qp)^N \tag{6}$$

The probability that two distinct selected rows are not equal is

$$P_{ne} = 1-P_e = 1-(1-2qp)^N \tag{7}$$

Define the probability P_{cr} to be the probability that for any two distinct selected rows, row i and row j, each element in row i equals or exceeds the corresponding element in row j through r elements. In this case, row i is said to contain row j through r elements. As above

$$P_{cr} = P_{cr-1}(1-qp) \tag{8}$$

$$P_{cr} = (1-qp)^r \tag{9}$$

The probability that row i fully contains row j (through all N elements) is P_c, where

$$P_c = P_{cN} = (1-qp)^N \tag{10}$$

It will also be useful to define the probability that one given row does not contain another given row as

$$P_{nc} = 1-P_c = 1-(1-qp)^N \tag{11}$$

To proceed, it would be useful to have formulas giving the probabilities that row i is not equal to any of a set of K other rows (distinct from each other and from row i), and that row i is not contained by any of a set of K other rows. It might at first be thought that, since the distinct rows j and k are independent of one another, the event that row i does not equal row j would be independent of the event that row i does not equal row k, with a similar statement where “is not equal to” is replaced by “is not contained by”. This is not, in general, the case, and it is worth the effort to fully understand why. If such were the case, then the probability that row i is not equal to any of a set of K other rows would be given by

$$P_{neK} = P_{ne}^{K} \quad \text{(incorrect)} \tag{12}$$

and the probability that row i is not contained by any of a set of K other rows would be given by

$$P_{ncK} = P_{nc}^{K} \quad \text{(incorrect)} \tag{13}$$

Suppose that the probability p is smaller than one-half, so that rows with fewer 1s are more likely than rows with more 1s. If it is given that row i does not equal row j, then there is an increased likelihood that row i is one of those less likely rows with more ones, and a correspondingly increased probability that row i does not equal some other row k. That is, the events that row i does not equal row j, and row i does not equal row k are correlated, even though rows j and k are independent of one another. The correlation is provided by the common row i in each comparison. If the probability p is exactly one-half, then any of the possible 2^N rows are equally likely, and the generally incorrect formula above for the probability that row i is not equal to any of a set of K other rows becomes correct.

When the "is not contained by" case is considered, the formula above is incorrect even when the probability p is one-half. Because rows with fewer 1s are more easily contained by other rows than are rows with more 1s, if it is given that row i is not contained by row j, then there is an increased likelihood that row i is one of those rows with more 1s, and a correspondingly increased probability that row i is not contained by some other row k.

If the above arguments are not sufficient to convince you of their merit, as they were unable to fully convince me, then a good, if tedious, way to convince yourself is by a full examination of the most simple case that presents the possibilities.

When there are three rows and two columns in the user request matrix, there are six elements and a total of 64 possible matrices. These matrices may be divided into seven groups, depending on the total number of 1s in the matrix: there is one matrix with no 1s, whose probability of occurrence is q^6; six matrices with a single 1, each of whose probability of occurrence is pq^5; fifteen matrices with two 1s, each of whose probability of occurrence is p^2q^4; twenty matrices with three 1s, each of whose probability of occurrence is p^3q^3; fifteen matrices with four 1s, each of whose probability of occurrence is p^4q^2; six matrices with five 1s, each of whose probability of occurrence is p^5q; and, finally, one matrix with six 1s, whose probability of occurrence is p^6.

Now, by counting cases, multiplying by the probability of occurrence, and adding, the probability that row 1 equals row 2 is given by

$$\begin{aligned} P(\text{row } 1 = \text{row } 2) &= q^6 + 2pq^5 + 3p^2q^4 + 4p^3q^3 + 3p^4q^2 + 2p^5q + p^6 \\ &= 1 - 4p + 8p^2 - 8p^3 + 4p^4 = (1-2pq)^2 = P_e \end{aligned} \tag{14}$$

which is just what is expected. Note

$$P_{ne} = (1 - P_e) = 4p(1 - 2p + 2p^2 - p^3) \tag{15}$$

Using the same method

$$\begin{aligned} P(\text{row } 1 \neq \text{row } 2, \text{row } 1 \neq \text{row } 3) &= 2pq^5 + 9p^2q^4 + 14p^3q^3 + 9p^4q^2 + 2p^5q \\ &= p(2 - p - 2p^2 + p^3) \neq P_{ne}^2 \end{aligned} \tag{16}$$

Figure 3 graphs the probability that row 1 is not equal to either row 2 or row 3, graphs what this value would be if the two events were independent, and graphs the difference. The two curves are reasonably alike, but this is for a very simple case, where one row is being tested for equality against two others. If the number of rows K in the set that is being compared to the single row i is larger than two, then the difference between the true result and the result derived from the assumption of independence will likely be larger.

For the probability that one row contains another

$$\begin{aligned} P(\text{row 2 covers row 1}) &= q^6 + 4pq^5 + 8p^2q^4 + 10p^3q^3 + 8p^4q^2 + 4p^5q + p^6 \\ &= 1 - 2p + 3p^2 - 2p^3 + p^4 = (1-pq)^2 = P_c \end{aligned} \tag{17}$$

which is just what is expected. Note

$$P_{nc} = (1 - P_c) = p(2 - 3p + 2p^2 - p^3) \tag{18}$$

By counting, multiplying, and adding as before

$$\begin{aligned} P(\text{row 1 is not covered by row 2 or row 3}) &= 2pq^5 + 5p^2q^4 + 6p^3q^3 + 4p^4q^2 \\ &= p(2 - 5p + 6p^2 - 4p^3 + p^5) \neq P_{nc}^2 \end{aligned} \tag{19}$$

There is something notable about this formula. All the prior probability formulas were symmetrical about the point $p = 1/2$, but this one is not.

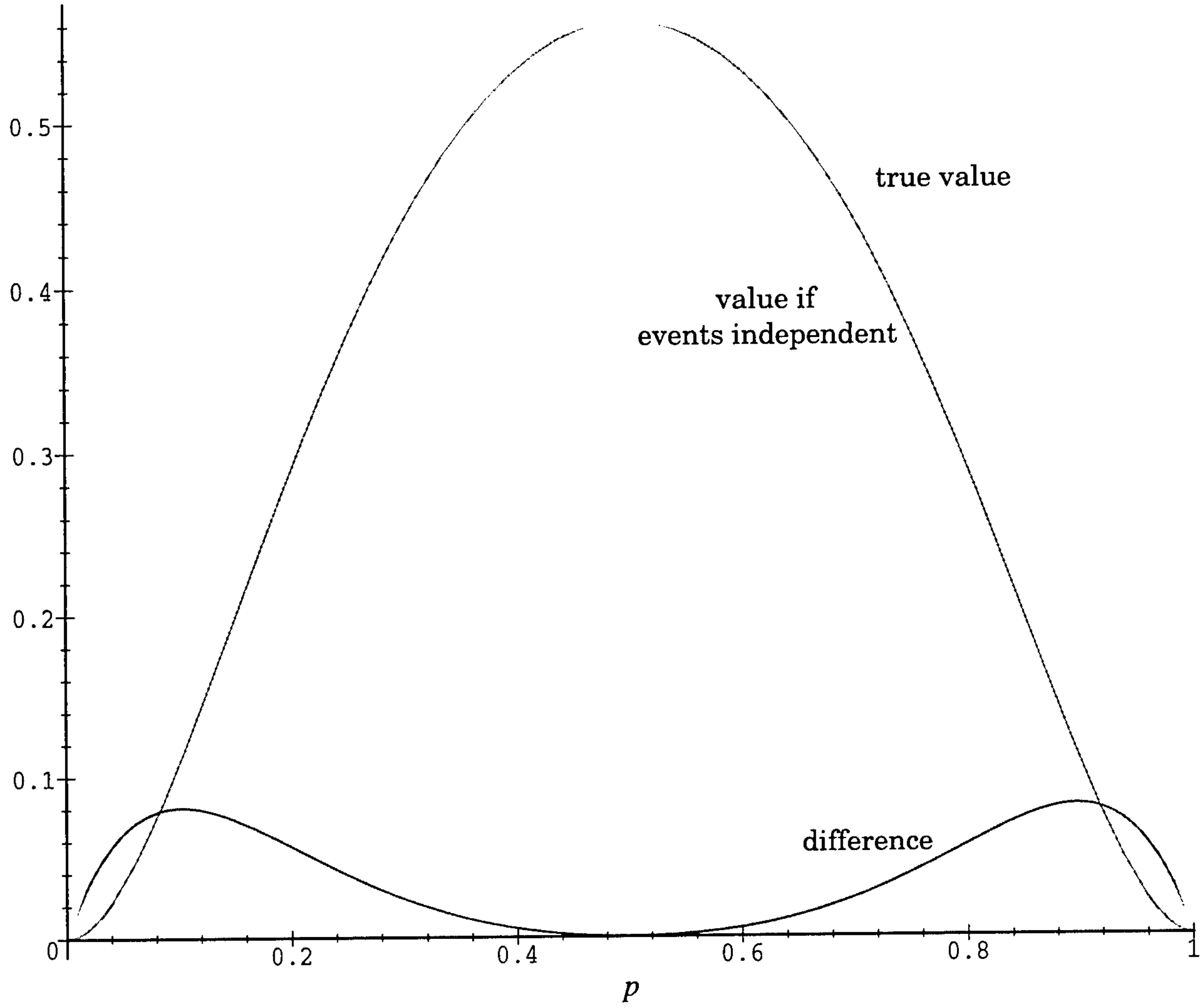


Figure 3: Probability that row 1 is not equal to either row 2 or row 3

Figure 4 graphs the probability that row 1 is not contained by either row 2 or row 3, graphs what this value would be if the two events were independent, and graphs the difference. In the case of containment, even in this simplest of examples, the difference between the two results is considerable. The difference would get larger if the one row were being considered for containment by more than two other rows.

These problems with event correlations have precluded the development of closed-form formulas to predict the performance of the containment algorithm and its relatives.

There are a few things that may be done with formulas. Comparing independently selected rows to a specified pattern for equality is a sequence of independent events. As the probability p becomes large, while the number of information regions is reasonably small and the number of users is large, there is an increasing likelihood that some user wants all the information regions. That is, there is an increasing probability that there is a row of all 1s in the user request matrix. This probability is easily formulated as.

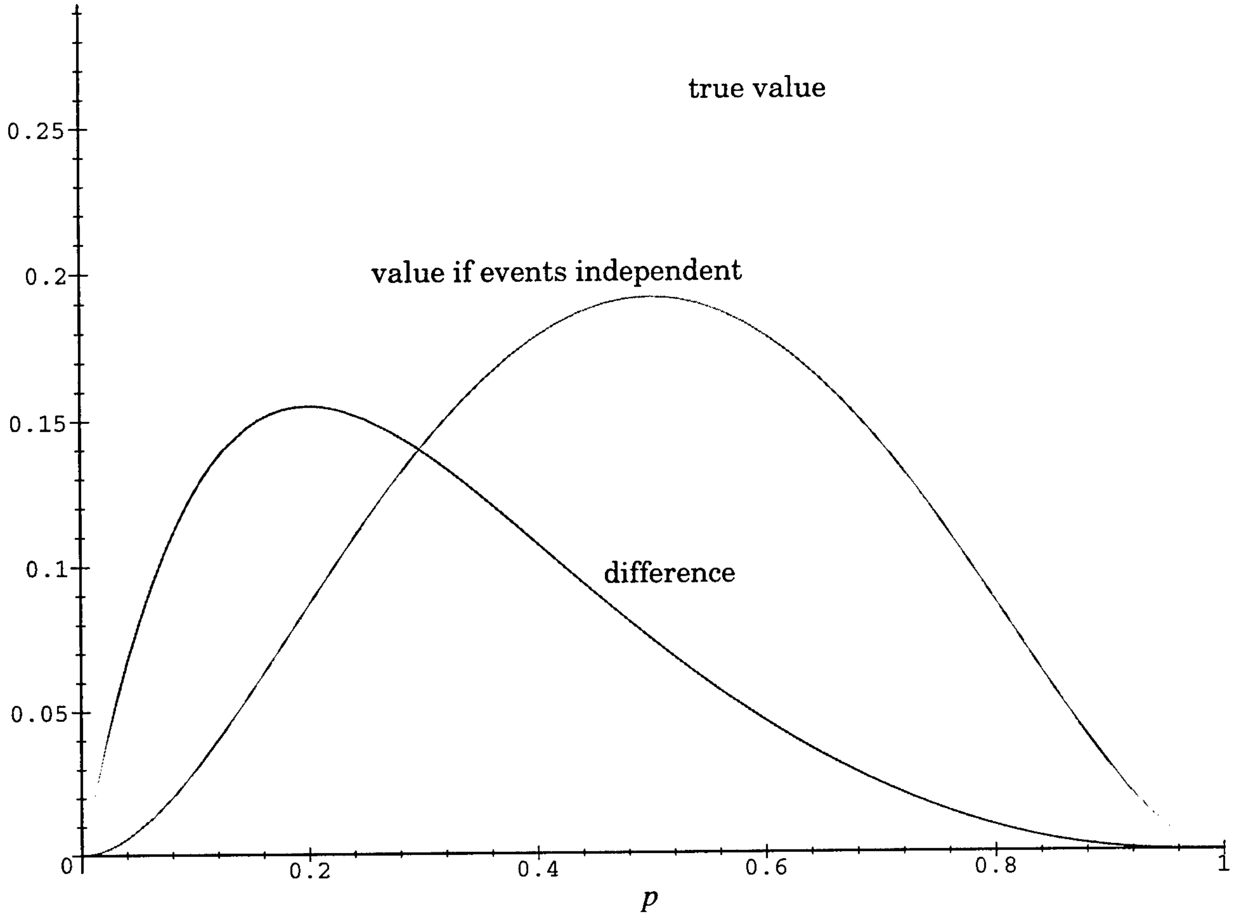


Figure 4: Probability that row 1 is not contained by either row 2 or row 3

$$P_{1s} = 1-(1-p^N)^M \tag{20}$$

A better way to examine this formula is to ask how many users would be required to make this probability equal 99%, as a function of N and p. The answer is

$$M_{1s} = \frac{\ln(0.99)}{\ln(1-p^N)} \tag{21}$$

Figure5graphsthisfunction,whichhelpsinunderstandinghowthecontainmentalgorithm behaves for higher values of the probability p. Given a value of p equal to one of those lines on the graph, if the number of users and the number of information regions corresponds to a point to the left of the line, then the containment algorithm will very likely produce the simple result that there is just one uncontained user who wants all the information regions. The behavior of the algorithm for small values of the probability p is more subtle

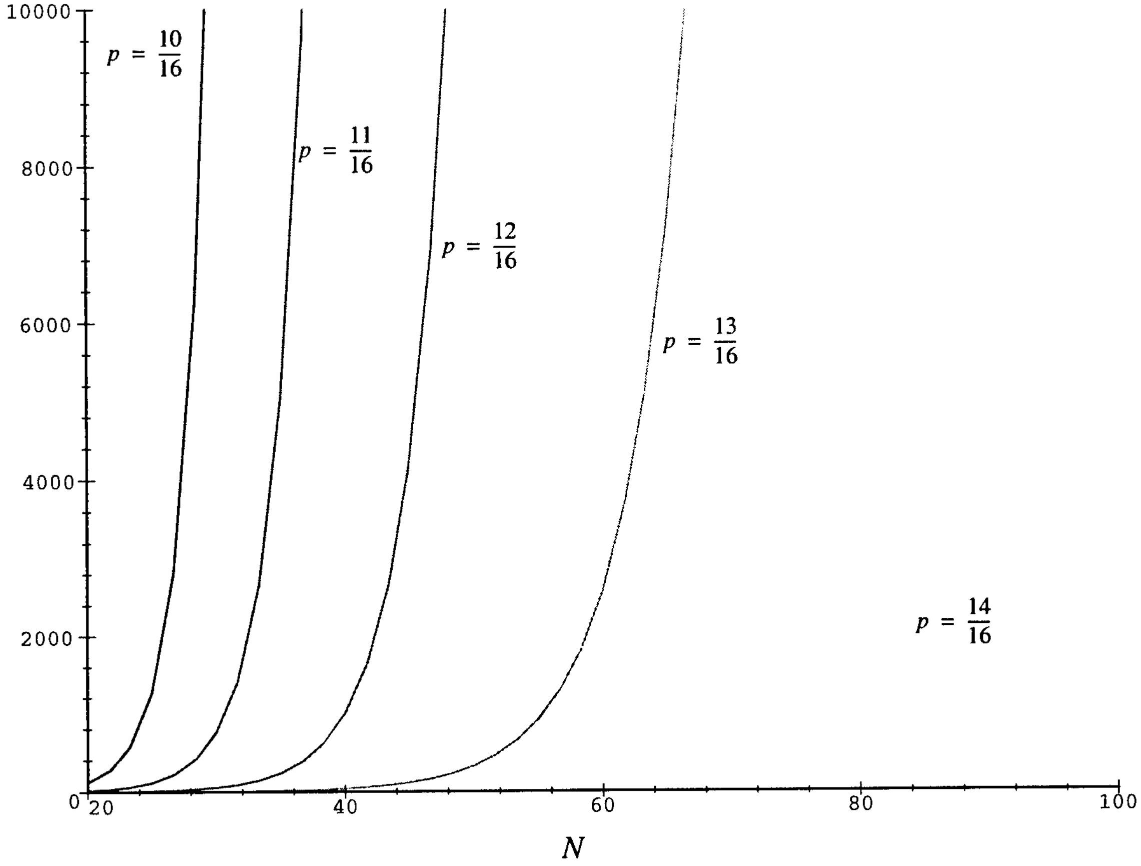


Figure 5: Number of users needed to make the probability of having a row of all 1s exceed 99%

An Algorithm Test Bed. A computer program was written to actually fill matrices with 1s and 0s, implement the containment algorithm, and count the number of uncontained rows. This program was written primarily to examine the potential utility of the containment algorithm, but there are other reasons. There is a need for a test bed for algorithm development to see just how a containment algorithm would be implemented efficiently and to see how much computer work is involved. Extensions or modifications of the containment algorithm and other algorithms entirely may be developed which will also need analysis, and the test bed can be quickly modified to provide it. The program has two modes of operation: a deterministic mode and a Monte-Carlo mode (it actually has several Monte-Carlo modes). If the total number of elements in the matrix (the number of users times the number of information regions) is quite small, say fewer than 28 or so, then there are only two raised to the power of this number of possibilities. With 28 elements, there would be just over 268 million different possible matrices. In the deterministic mode, each of these possibilities is examined in turn. The probability of a particular matrix occurring is easily computable by just counting the total number of 1s in the matrix, and applying the containment algorithm to each matrix allows the number of uncontained rows and related statistics to be computed.

Finally, the probability-weighted sum of the number of uncontained rows is calculated as the expected number of uncontained rows. The deterministic mode allows many different probabilities p to be examined at once. The same deterministic sequence of matrices with the same numbers of uncontained rows are involved, regardless of p. Only the weighting factors involved in computing the expected number of uncontained rows change as p changes, and this is a small part of the algorithm in time or space

Prime interest in the algorithms is on cases with many more than 28 elements in the matrix, in fact, with thousands of elements in the matrix, where the deterministic mode is out of the question. The Monte-Carlo mode fills a matrix according to a specified probability that any element is 1, then applies the containment algorithm to count the number of uncontained rows, and repeats this process for a specified number of samples—generally a very tiny fraction of the possible number of such matrices—and simply averages the results for an approximation to the expected number of uncontained rows. The Monte-Carlo mode can only consider a single value for the probability p at one time, since the elements in the sample matrices depend on this value. It takes a number of operations roughly proportional to M^2N to simply examine containment for a single matrix, so even the Monte-Carlo mode is very slow to handle matrices involving 1000 or more users and 100 or more information regions.

In order to make filling the matrix very efficient, the random fill algorithm at present only accommodates element probabilities that are integral multiples of one-sixteenth. This makes it easy to use an integer random number generator and use four bits of the random integer at a time to generate each element. The random integer generator is a simple multiplicative generator that always produces positive odd integers, so the first and last bits of the 32 are unusable. Seven elements of the matrix are generated from one random integer. At first, it was thought that this range of probabilities would suffice, but it may be that smaller probabilities than one-sixteenth need examination. If so, the random fill algorithm must be rewritten.

The Containment Algorithm Itself. The implementation of the algorithm is fairly simple. First, an array of integers, the containment vector conv[], is constructed with one element for each user, and each element of this array is initialized to -1. This value indicates an uncontained user request, and is temporary. When it is discovered that the request of user i contains that of user j, then conv[j] will be set to i, indicating both that user j is contained and by whom. Of course, it may later be discovered that the request of user i is itself contained by that of user k, so that conv[i] is set to k, and this process may produce containment chains that ultimately lead to an uncontained user request. After the containment algorithm is completed, the chains of containment indicated by the conv array are followed up so that each contained row is indicated, in conv, as being contained by some uncontained row, which is the end of the chain.

The algorithm loops over all the rows, calling the current row i. If row i is already contained (if conv[i] is not -1), then row i is not further considered. For each considered row i, the algorithm loops over all the remaining rows, beginning with row i+1, calling the current inner loop row j. If row j is already contained (if conv[j] is not -1), then row j is not further considered. Having specified rows i and j, two temporary logical variables, iconj and jconi, are set true temporarily. A test is made, going down the two rows one element at a time, to find rows that do not contain each other, since this is all that one element can tell. If lack of containment is found, then the appropriate logical variable is set false, and if both logical variables are false, the test for these two rows is terminated. After the test is complete, if row j does contain row i, then conv[i] is set to j, and the row i loop is advanced (which starts a new set of rows j). If row i contains row j, then conv[j] is set to i, and the next row j is considered. At the beginning, the algorithm sets the number of uncontained rows equal to the total number of rows, and then subtracts one whenever a contained row is found, keeping track of the number of uncontained rows.

Once the containment algorithm is done and the conv array chains are followed up, various related measures of the performance of the algorithm may be computed, such as the maximum number of information regions in any one uncontained row, and the total number of information regions in all the uncontained rows.

Results of the Simulation. A single (very long) run of the program in its deterministic mode examined every case where the product of the number of users and the number of information regions is 28 or less (46 possibilities), and for each value of the probability *p* that is an integer multiple of one-sixteenth, from one-sixteenth to fifteen-sixteenths. Table 1 shows a sampling of the simulation results from this run. The first three columns are the input parameters. The fourth column is the expected numbers of uncontained users. The "max regions" column is the expected maximum number of information regions in any one channel, while the "total regions" column is the expected number of information regions included in all the uncontained user requests, where some regions may be included more than once.

Interest in such small numbers of users and information regions is limited, but the simulation cannot practically treat larger numbers in a deterministic fashion. Examination of larger problems must be by Monte-Carlo methods.

Table 1: Deterministic Simulation Results Sample

rows *M*	cols *N*	prob *p*	uncontained users	max regions	total regions
3	2	0.0625	0.34038	0.33274	0.35205
3	2	0.0500	1.17188	1.56250	1.75000
3	2	0.9375	1.00129	1.99822	1.99951
3	9	0.0625	1.23402	1.15811	1.59834
3	9	0.0500	2.59857	5.75193	12.25212
3	9	0.9375	1.13423	8.91332	9.94293
6	4	0.0625	1.15521	0.91561	1.28872
6	4	0.0500	1.96416	3.21454	5.42680
6	4	0.9375	1.00030	3.99986	4.00076
9	3	0.0625	1.12290	0.92377	1.32455
9	3	0.0500	1.41386	2.69739	3.48926
9	3	0.9375	1.00000	3.00000	3.00000

A Monte-Carlo run of the simulation, sampling 100 random user request matrices for each size and probability considered, was made with the number of users from 1000 to 5000, the number of information regions from 20 to 50, and the probability p varying from one-sixteenth to fifteen-sixteenths. These ranges were determined to be interesting by intersecting the problem region of interest with the region where the algorithm produces any useful results. The problem interest centers about many more information regions than 50, but the utility of the algorithm degrades as the number of information regions increases above 50, unless the probability is less than the one-sixteenth value, which is the smallest value for which the simulation was run. Even for the ranges simulated, some of the results are uninteresting, especially when the probability becomes large.

Figure 6 shows a typical result of the number of uncontained users as a function of the probability p, for 2500 users and 30 information regions. As the probability increases, the number of uncontained users increases rapidly until a maximum is reached somewhere near a probability of one-half, then the number of uncontained users falls rapidly with increasing probability until only one user is uncontained, because it is likely that some user has requested all the information regions (as Figure 5 above would indicate). The maximum number of uncontained users is, in this example result, about four-fifths the total number of users.

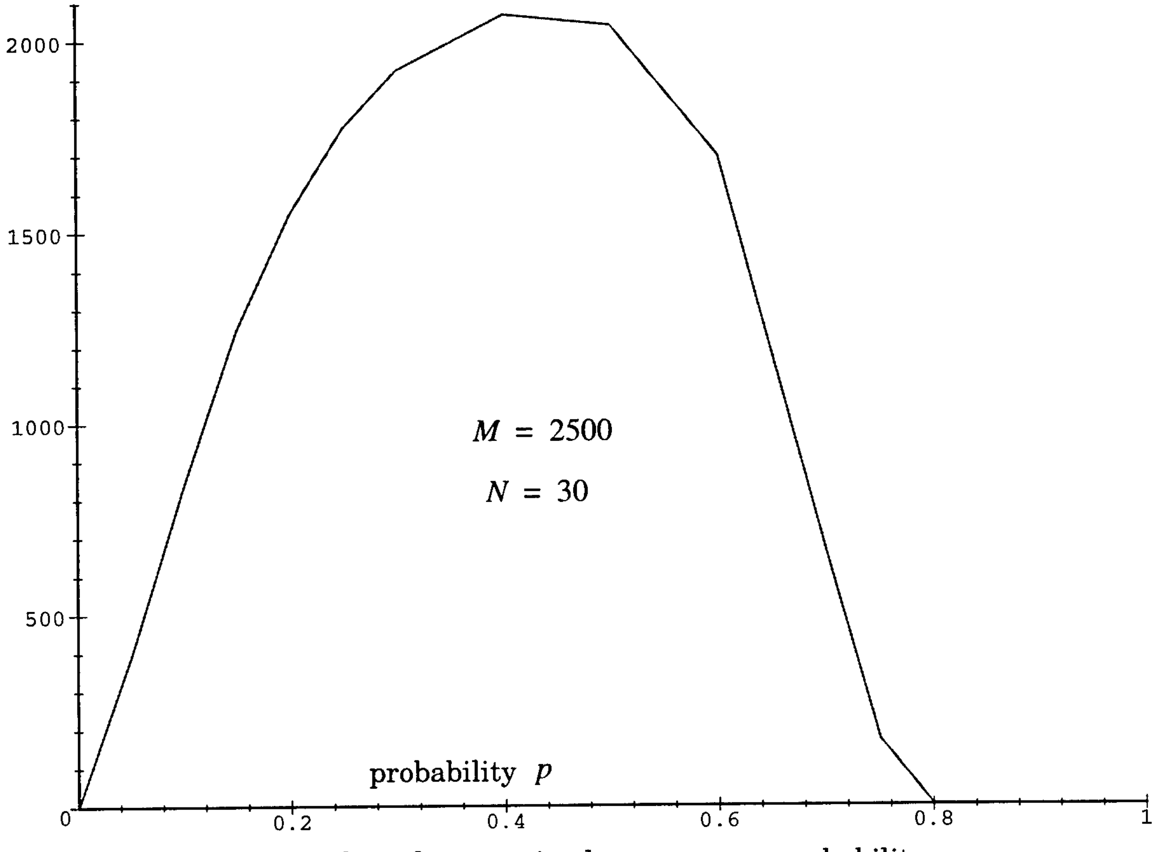


Figure 6: Number of uncontained users versus probability

Figure 7 shows the same kind of result, but for 5000 users and 50 information regions. What is illustrated is that the number of uncontained users is about equal to the total number of users over much of the center of the range for probability. Such behavior is not useful, so as the number of information regions increases, interest in the algorithm shifts to small values of the probability.

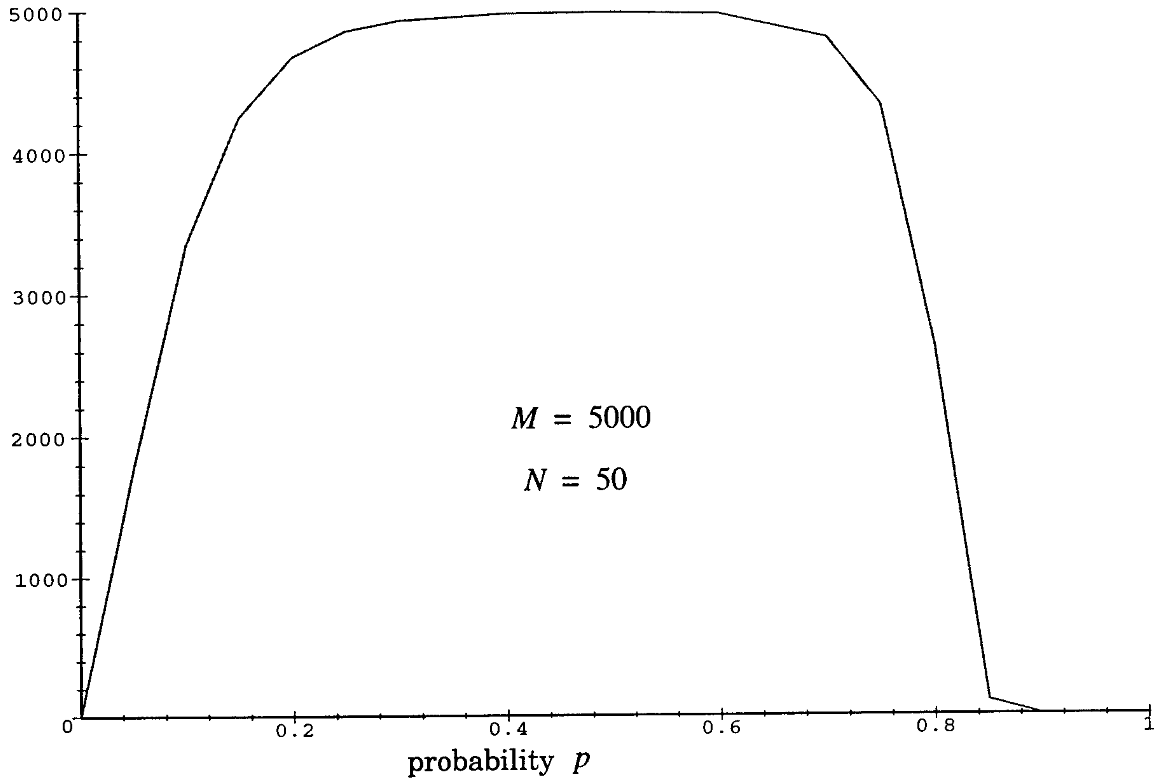


Figure 7: Number of uncontained users versus probability

Table 2 shows the average results of the containment algorithm on 100 randomly selected user request matrices when the probability p is 0.5. The rows are for the number of users, from 1000 to 5000, while the columns are for the number of information regions, from 20 to 50. The main entries are the expected number of uncontained user requests. As can be

Table 2: Monte-Carlo Simulation Results Sample; $p = 0.5$

M\N	20	25	30	35	40	45	50
1000	294	627	844	936	977	997	999
1500	342	844	1202	1365	1452	1493	1498
2000	379	1008	1529	1789	1920	1986	1996
2500	411	1156	1832	2186	2380	2479	2494
3000	383	1256	2115	2584	2832	2971	2991
4000	406	1489	2663	3319	3720	3951	3985
5000	400	1633	3159	4034	4588	4926	4976

seen, when there are forty or more information regions, there is little or no containing of user requests over the range of users examined. This could be expected. With forty regions, there are two to the fortieth power possible user requests, which is more than ten to the twelfth power, and, with an element probability of one-half, each of these request rows is equally likely. There is very little chance that one row will contain another.

Table 3 shows the same results when the probability p that any individual user wants any individual information region is reduced to 0.0625 (one-sixteenth) Here, the containment algorithm shows considerable merit, but is of little help when the number of regions increases.

Table 3: Monte-Carlo Simulation Results Sample; $p = 0.0625$

M\N	20	25	30	35	40	45	50
1000	95	147	206	235	329	405	470
1500	117	189	273	314	454	563	662
2000	139	229	332	390	565	717	841
2500	161	266	394	452	677	859	1018
3000	181	304	452	511	774	994	1182
4000	214	364	546	611	955	1238	1487
5000	241	416	628	700	1128	1458	1764

Since major interest is on situations with up to a thousand information regions, it might be thought that the containment algorithm is simply unusable, but that is not necessarily true. User requests in the "real world" are not purely random, and it is not a requirement that only one channel of information regions be sent to each user. It is entirely possible that the totality of information regions may be divided up into groups of 25 or 50 regions (on the basis of geographic region of origin perhaps, or some other natural affinity), and each group of regions is either very likely or very unlikely to be wanted by a user (depending on the geographic region of interest of the user). After grouping, the containment algorithm may provide significant help in preparing channels that consider only one group of information regions, with possibly several such channels (for differing groups) being sent to a user.

Zipf Model Results. The Monte-Carlo results above are based on a model where each binary element of the user request matrix is statistically independent of the others, and each has the same probability of being 1. Requests for information are a kind of popularity contest, and it is usual in such contests that a few of the options are much more popular than are the others. A second probability model was used to consider such a case. It is called the Zipf distribution, named after a Harvard linguist who used it to model the frequency of occurrence of words in English text. In the Zipf model, if the options—here, the information regions—are ranked according to their popularity, from 1 to N, then the probability that the nth region will be requested by any given user is

$$p_n = \frac{c}{n^{\beta}} \tag{22}$$

where c is a constant between 0.0 and 1.0, and the exponent β is near 1.0 (from 0.5 to 2.0 in this investigation).

Using this model, the total number of 1s in any one row of the user request matrix is expected to be

$$E\{\text{\# of 1s per row}\} = c \sum_{n=1}^{N} \frac{1}{n^{\beta}} \tag{23}$$

If this value is divided by the row length N, it becomes the "average" probability of selecting any information region. Figure 8 plots the average, when the constant c is 1.0, for three values of the exponent β. The utility of this plot is that it provides some basis for comparison of the Zipf results to the Monte-Carlo results above, where all requests are equally likely.

Figure 9 illustrates some of the Zipf results for 2500 users and 30 information regions.

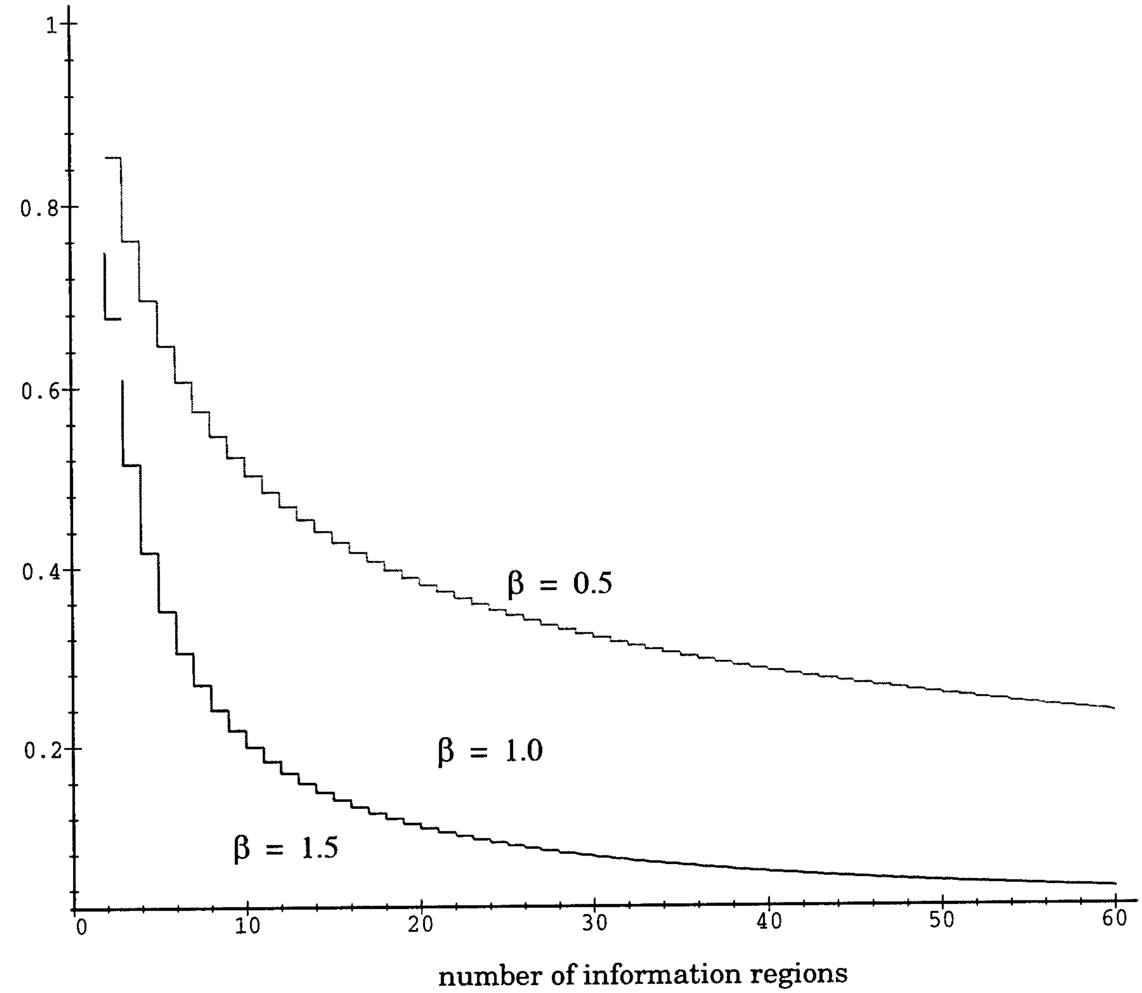


Figure 8: Zipf model overall average probability of requesting a region

containment algorithm

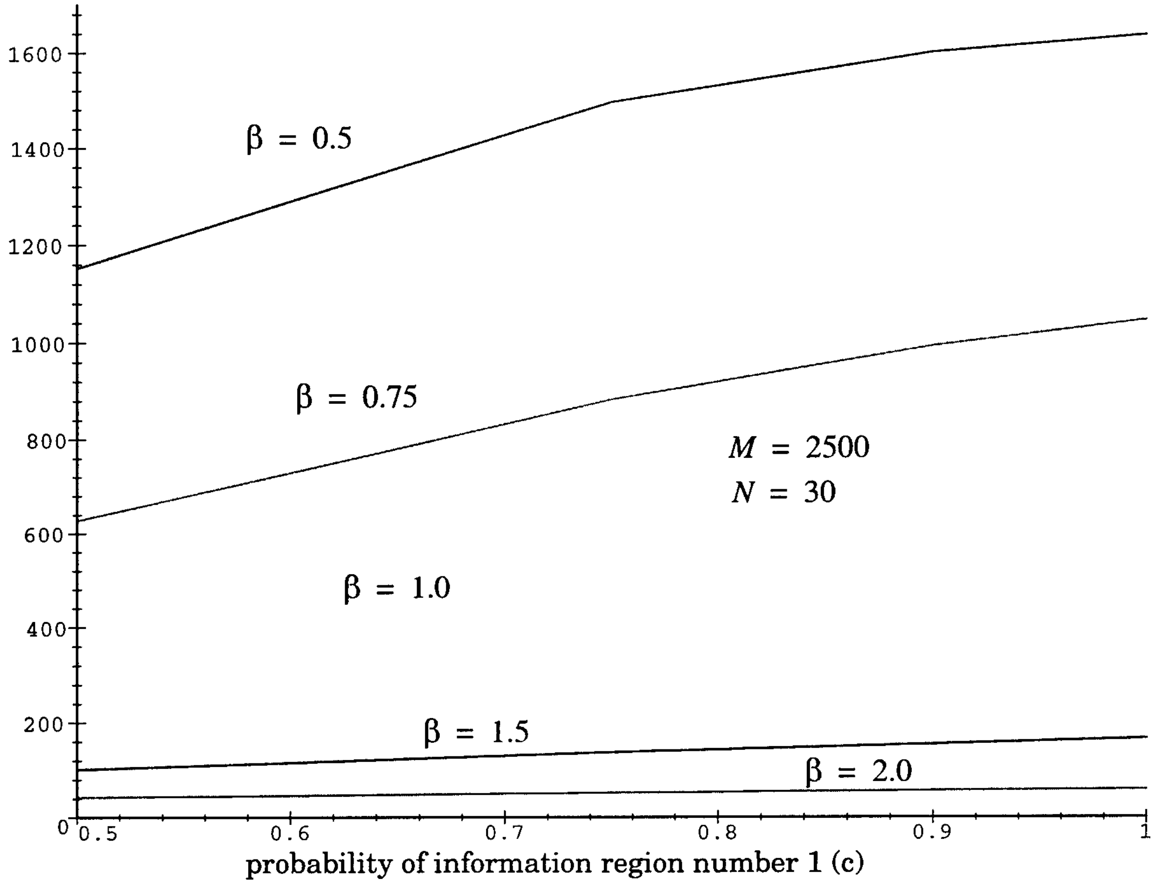


Figure 9: Number of uncontained users versus probability (Zipf model)

Comparison with Simulation. The simple testbed simulation whose results are presented in this paper is not the main ChannelTech software. The containment algorithm was also tested in the main software, but it is not easy to compare those results to the testbed results, because the probabilistic model used to generate user requests differs between the two.

In the main simulation, information regions are described by attributes. In the main test runs discussed here, called "the special case", only three attributes are relevant: data type, access method, and keyword. The data type is one of the three list items: image, text, or video. The access method is one of the two list items: HTTP or FTP. The keyword is one of the 26 letters of the alphabet. An information region is described by a single data type, a single access method, and a single keyword, so there are 3x2x26 = 156 data regions. A user request specifies a single data type or all three data types, a single access method or both access methods, and a single keyword. There are only 4x3x26 = 312 possible different user requests. A user request may include 1, 2, 3, or 6 data regions. For each keyword, there are 12 possible different user requests, as tabulated below in Table 4.

Table 4: User Requests (Each Keyword)

category	image	text	video	HTTP	FTP
1	*			*	
2	*				*
3		*		*	
4		*			*
5			*	*	
6			*		*
7	*	*	*	*	
8	*	*	*		*
9	*			*	*
10		*		*	*
11			*	*	*
12	*	*	*	*	*

In Table 4, request category 12 is regarded as being at the top containment level—it contains all the other requests in the Table. At the middle containment level are request categories 7 through 11—none of these contains any of the others, although each contains two or three of the requests at the lowest level. At the lowest containment level are the request categories 1 thorugh 6—none of these contains any of the others. For each keyword, it is easy to see that there are a maximum of six uncontained user requests (those at the lowest level). Any additional user requests would reduce the number of uncontained requests. Thus, in toto, there are a maximum of 6x26 = 156 uncontained user requests (which happens to be the same as the number of information regions).

When user requests begin to come in, the number of uncontained user requests will, at first, grow to something approaching 156. As the number of user requests becomes large, for each keyword requests of category 12 will occur, and the total number of uncontained requests will tend toward 26—one category 12 request for each keyword.

Each of the request categories tabulated above is equally likely to be selected in the main software. This means that each information region has an equal probability of being selected in any user request, a probability of 1/78. However, these selections are not independent. Given that a region of any specific keyword is chosen, the probability of also choosing a region of a different keyword is zero, not 1/78.

When the testbed software is run with 156 information regions and with a probability of selecting any individual region in a single user request of $p = 1/78$, the results differ markedly from the main software tests. In the testbed, there are 2 to the 156th power possible different user requests, not 312, and equality and containment of requests is far less likely. As a consequence, the number of uncontained user requests is not limited, but grows rapidly with a growing number of total user requests. These results were obtained on the first trial of the testbed program, and are shown in Figure 10.

In an attempt to reproduce the main software test results for the special case, the testbed program was altered to fill the user request matrix with random 1s and 0s according to the actual special case rules. The testbed software results for this second trial are shown in Figure 11. These results are what is expected. The slight lump in the curve at 900 total users is a statistical anomaly. Repeated runs of this case (averaging over 100 cases) are more inline with the rest of the curve.

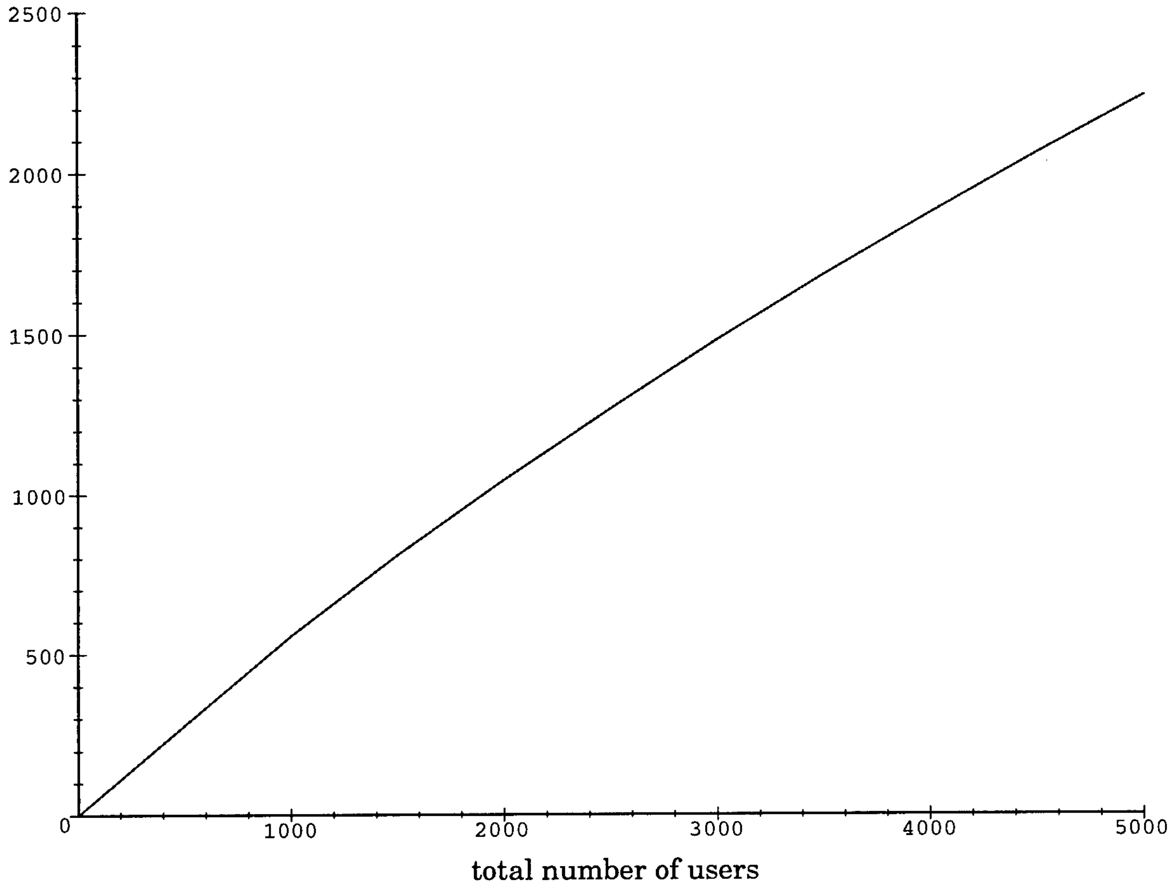


Figure 10: Uncontained users vs users for the special case (Trial 1)

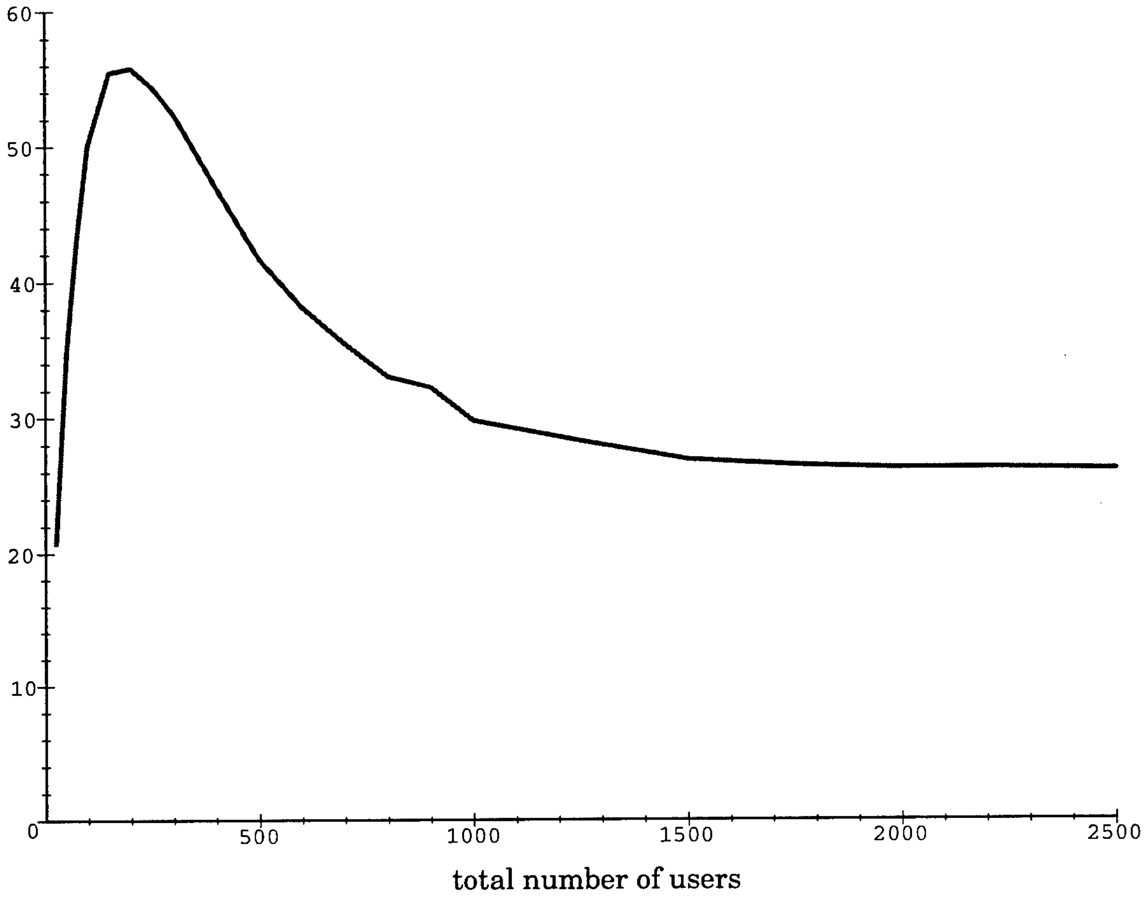


Figure 11: Uncontained users vs users for the special case (Trial 2)

Extensions to the Containment Algorithm—Clustering. It might be possible to alter the containment algorithm somewhat and improve its performance. One conceptual alteration is to contain two or more nearly equal user requests with a new, phantom, user request that contains them all, thus clustering several requests (that do not necessarily contain one another) into one. This might be done before, after, or in place of the containment algorithm.

The concept of "nearly equal" is based on the Hamming distance between two binary sequences, which is simply the number of positions in which they differ. This measure, applied to the rows of the user request matrix, may be used to decide if two requests are nearly equal.

The concept of containment leads almost immediately to an algorithm. One reason this happens is that the concept of containment is transitive. If row i contains row j, and row j contains row k, then row i contains row k. It is the property of transitivity that leads to containment chains. In the containment algorithm, no user request rows are ever altered, and it makes no difference (except possibly for efficiency) in what order the user requests are considered, the results are the same.

The concept of Hamming distance is not transitive. If row i is near row j, and row j is near row k, then row i may not be near row k. This makes it hard to find groups of rows that are all contained by some nearby (phantom) user request. It is easy to imagine a sequence of rows, each near its immediate neighbors but not near any other row in the sequence.

The results of replacing rows by phantom rows depends on the order of operations. Suppose row 0 and row 1 have a Hamming distance of two, and row 0 is replaced by the phantom row that contains both, while row 1 is marked as being contained by the revised row 0. It might have been that row 2 had a Hamming distance of two from the original row 0 as well, but it will not have a Hamming distance of two from the revised row 0. Or row 2 might have a Hamming distance of two from the revised row 0, where it did not so nearly approach the original row 0.

In the clustering algorithm, as implemented in the simulation, pairs of (uncontained) rows are considered, with row i being in an outer loop that starts at the top of the user request matrix, and row j being in an inner loop that starts with the next row after i. When the two considered rows are within the specified Hamming distance of one another, row i is replaced by the new containing row; row j is marked as contained by row i; no more rows j are considered for this row i; and row i is advanced by one until there are no more rows to consider. When the clustering algorithm completes, there may still be rows within the maximum Hamming distance of one another. The algorithm could be repeated, with possible further reductions in the number of uncontained rows..Consider first using clustering after the containment algorithm. Once the containment algothm has eliminated from consideration the contained rows of the matrix, there are no two equal rows left, rows whose Hamming distance from one another is zero. Nor are there any two rows left whose Hamming distance is one, for, if there were, one of those rows would have a 1 where the other had a 0, being otherwise the same, and thus contain the other row. So the minimum Hamming distance among the uncontained rows is two.

Reverse Containment. The concept of containment may be used for a different kind of algorithm. Begin with any channelization and user subscription matrices. For example, start with the one-channel-per-user approach. Then search for channels that are contained by other channels. When channel i is contained by channel j, remove row i from row j of the channelization matrix C (remove all the 1s in row i from row j), and then go down column i of the subscription matrix S, placing 1s as necessary to assure that each user previously subscribing to channel j now subscribes to both channels i and j. That is, once i and j are found, for $n = 1, \ldots, N$, set C_{jn} to 0 whenever C_{in} is 1; then for $m = 1, \ldots, M$, set S_{mi} to 1 whenever S_{mj} is 1.

This reverse containment algorithm does not, at first, appear to change the number of channels. It simplifies some channels by breaking them into parts, which are then reassembled in the user subscriptions, causing the total system information load to decrease, while not increasing any user load. In fact, the number of channels may decrease, because all information regions may be removed from them. If one channel happened to be the union of two others, then the two smaller channels could be removed from the larger one, leaving nothing.

This algorithm, too, is not completely defined by the description above. Row k may contain both rows i and j (and others). Which, of these contained rows, is removed from row k is a matter of choice, and here it is not easy to see which choice is best. Once a containing channel has been made smaller by removal of a contained channel, then the new smaller channel is likely to be contained by others, and it may be removed from them.

When this algorithm is tested on randomly constructed problems that have fewer information regions than users, especially if the probability that a given user requests a given information region is either low or high, what often happens is that the channelization is initialized to the one-channel-per-user solution, but winds up at the one-channel-per-information-region result—a result having fewer channels and far lower total system load. When there are many more information regions than there are users, the number of containments becomes vanishingly small, and the reverse containment algorithm does nothing.

As implemented in the algorithm test bed, the reverse containment algorithm considers the rows of the channelization matrix, not in their storage order, but in order of their length—length being the number of 1s in the row—from shortest to longest. This means that the shortest (non zero) rows are removed from the others that contain them first. It also means that the process must be repeated, because the removal of one row from another shortens one row. For 5000 users and 50 information regions, this process can take minutes on a PC, so that averaging over 100 trials can take hours for a single probability, a single number of users, and a single number of information products. And the answer is almost always the same when, as in this case, there are many more users than there are information regions. As mentioned above, the result is that one channel is assigned to each information region alone.

CHANNELIZATION

Introduction

This note summarizes terminology and preliminary thoughts pertinent to the problem of channelizing information to be distributed over a network. There are presumed to be multiple users who want various portions of a large spectrum of information. The users indicate their information interests by submitting information profiles to a central distribution point, where some strategy is devised to disseminate the information. The distribution strategy is to compose information into various channels and then distribute the channels, having users subscribe to those channels containing the information they want. Many ways have been used to describe this problem, and it can be confusing to compare ideas with similar, but not identical, terms and concepts.

The discussion here veers between a clean, mathematical problem description and consideration of more practical, worldly issues.

Users and Information Regions

One way to begin to think about the channelization problem is to specify a number of users and a number of disjoint, elementary information regions.

M is the number of users of information.

N is the number of regions of information, sometimes called information products.

A is the user information request matrix, an $M \times N$ binary matrix.

$$\langle A \rangle_{mn} = \begin{cases} 1 & \text{if user } m \text{ wants information region } n \\ 0 & \text{if not} \end{cases} \tag{1}$$

Generally, this user request matrix is considered as known.

Profiles and Factoring

In practice, information is described by attributes, a predefined list of parameterized features. For example, one attribute of information might be the geographic location to which it pertains. The parameters of this attribute might be latitude and longitude limits of a region of the Earth. Another attribute might be information type, with the parameter being another list item, such as imagery, signal, or communications. A user does not generally have knowledge of the existence of any particular item of information. Instead, a user requests information that matches a submitted information profile, where the attributes of the desired information are listed.

A profile set consists of all information that matches a certain user request profile. Information regions are the disjoint, nonempty intersections of the profile sets and their complements, so that each user profile set is simply the union of some such regions. Since there are M user profiles and corresponding profile sets, there could be as many as $2^M - 1$ information regions.

Suppose the subsets of information that match the user profiles are denoted by P_m, for $m = 1, \ldots, M$. From these profile sets, the information regions are obtained by a process called factoring. In what is termed "fully-factored" form, the information regions could be denoted by R_n and numbered from 1 to $2^M - 1$, and

$$R_n = \bigcap_{m=1}^{M} Q_{nm}, \tag{2}$$

where the binary representation of n is $b_{nM}b_{nM-1}\ldots b_{n2}b_{n1}$, and

$$Q_{nm} = \begin{cases} P_m & \text{if } b_{nm} = 1 \\ \overline{P_m} & \text{if } b_{nm} = 0 \end{cases} \tag{3}$$

The profile sets may be recovered from the regions by

$$P_m = \bigcup_{n=1}^{2^M - 1} T_{mn} \tag{4}$$

where

$$T_{mn} = \begin{cases} R_n & \text{if } b_{nm} = 1 \\ \varnothing & \text{if } b_{nm} = 0 \end{cases} \tag{5}$$

Another way to put this, perhaps easier to grasp, is to express the elements of the user request matrix as

$$A_{mn} = b_{nm} \tag{6}$$

For example, if $M = 4$, then $R_{10} = P_4\overline{P_3}P_2\overline{P_1}$, because $10 = 1010)_b$. The complete user request matrix in this fully-factored example case is

$$A = \begin{bmatrix} 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \end{bmatrix} \tag{7}$$

Note that the nth column of A, reading from bottom to top, has the binary digits of n.

Many, perhaps most, of these fundamental information regions may be empty, and may be disregarded. That is, numerous columns may be missing from the fully-factored form of the request matrix, because they correspond to empty information regions. It will not be assumed in the sequel, unless made explicit, that the information regions correspond to the user requests in this fully-factored form.

Information Attribute Parameter Ranges

It can be very difficult to factor user profiles into information regions when the information attributes are parameterized by other than finite lists of items. For a prime example, suppose one of the attributes is geographic location, as specified by arbitrary ranges of latitude and longitude. Then, in any one user profile, the geographic range of information corresponds to the union of rectangles in latitude-longitude coordinates, perhaps only one or two such rectangles. If many such profiles are factored into information regions, the geographic ranges for the regions may quickly devolve into complex figures in latitude-longitude coordinates. Each of these figures could still be represented as the union of (very many, tiny) rectangles, but the bookkeeping would be demanding. It is probably best to treat even those attributes that are naturally described by arbitrary continuous ranges of some parameter or parameters as a finite list instead, specifying geographic regions by UTM map squares, for example, or even by much larger natural regions of interest.

Such partitioning of continuous variable ranges into finite lists enables a much simplified factoring process, and more. One attack on the channelization problem is to recognize when user requests or channel contents differ by little, so two or more requests may effectively be considered as one. That is, a "blurring" of the user profiles is somehow accomplished to identify "look-alike" profiles. Going from arbitrary continuous parameter ranges to preestablished lists of parameter ranges is essentially a step in the blurring process, one that regularizes requests so that look-alikes are easier to identify.

Profile Sets Versus Information Regions

A significant issue is that of beginning the consideration of the channelization problem with the concept of information regions at all. Given numerous user profiles, the problem of constructing the potentially much more numerous information regions can be so difficult that it will never be practically accomplished. A mere 500 users could generate 10^{150} information regions, a completely impractical number to deal with, even if most of the regions turn out to be empty. It is very likely that any practical algorithms will have to deal with the profile sets themselves, and not presume that they are first factored into disjoint regions.

Most of the thinking about the channelization problem, however, has begun with the starting point of information regions, and some of that thinking is summarized here, so the regions viewpoint will be presented.

Channels

K is the number of channels of information.

C is the channel information matrix, or channelization matrix, a $K \times N$ binary matrix.

$$\langle C \rangle_{kn} = \begin{cases} 1 & \text{if channel } k \text{ contains information region } n \\ 0 & \text{if not} \end{cases} \tag{8}$$

S is the user channel subscription matrix, an $M \times K$ binary matrix.

$$\langle S \rangle_{mk} = \begin{cases} 1 & \text{if user } m \text{ subscribes to channel } k \\ 0 & \text{if not} \end{cases} \tag{9}$$

Generally, the number of channels, the channelization matrix, and the subscription matrix are considered as unknown.

Note that the matrix product SC is an $M \times N$ matrix of integers where the mnth element is the number of times user m receives information region n. If each user is to get all the information that is requested, then $SC \geq A$. This constraint, unfortunately, is a nonlinear one if both C and S are unknowns.

The nonlinearity of the constraint above has been a serious block to attempts to cast the channelization problem into what is called a “mixed-integer, linear program”. Many scheduling and resource allocation problems that have similarities with the channelization problem may be so cast, and there is a wealth of study results, polished algorithms, and software available to attack such problems. It is attractive to think that the channelization problem might benefit from these studies, but it has not, as yet.

The function $\text{bin}(\ )$ is defined as

$$\text{bin}(x) = \begin{cases} 1 & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases} \tag{10}$$

R is the user information receipt matrix, an $M \times N$ binary matrix, defined element-by-element as

$$\langle R \rangle_{mn} = \mathrm{bin}(\langle SC \rangle_{mn}) = \begin{cases} 1 & \text{if user } m \text{ receives region } n \\ 0 & \text{if not} \end{cases} \tag{11}$$

or, in shorthand, as $R = \mathrm{bin}(SC)$.

Information Density

D is the information density matrix (vector), an $N \times 1$ nonnegative matrix whose nth element measures the density of information region n, and is generally a known quantity. A crude approximation when no density information is available is to treat all nonempty information regions as having unit density, that is, to set $D = 1_N$, where 1_N is the $N \times 1$ matrix (vector) with elements all 1s.

In practice, what is given, if anything, is not an information density vector for the regions, but some way to estimate the information density of a specified information set, be it a profile, a region, a channel, or some other set. In theory, applying this function to the regions produces the density vector, although that computation itself might be impractical.

The matrix product CD is the channel load matrix, a $K \times 1$ nonnegative matrix (vector) whose kth element is the total information density on channel k.

The scalar $1_K^T CD$ (which is simply the sum of all the elements of CD) is the total system information density, or the total system load—the sum of the information densities on all the channels.

The matrix product SCD is the user received load matrix, an $M \times 1$ nonnegative matrix (vector) whose mth element is the total information density received by user m.

The matrix product AD is the user requested load matrix, an $M \times 1$ nonnegative matrix (vector) whose mth element is the total information density requested by user m.

The matrix product $(SC - A)D$ is the user excess load matrix, an $M \times 1$ nonnegative matrix (vector) whose mth element is the total excess information density for user m. This definition depends on the assumption that the user gets all the requested information, so that $SC \geq A$.

There may be limits on information density. The system as a whole might have a limit, and each user might have a limit.

Utility

U is the user information utility matrix. It is an $M \times N$ nonnegative matrix whose

mnth element measures the utility to user m of information region n.

V is the user value matrix (vector). It is an $M \times 1$ nonnegative matrix whose mth element is the total utility of the information received by user m. Note that

$$\langle V \rangle_m = \sum_{n=1}^{N} \langle U \rangle_{mn} \langle R \rangle_{mn} \qquad (12)$$

This operation is not a standard matrix operation. The vector V is the main diagonal of the matrix UR^T, but the other elements of this matrix have no meaning.

V_T is the total system value, the sum of the values received by the users.

$$V_T = \sum_{m=1}^{M} \langle V \rangle_m = \mathrm{tr}(UR^T) \qquad (13)$$

where tr( ) is the matrix trace operator, the sum of the elements on the main diagonal.

Maximizing value may be an issue in some channelization problems.

The Channelization Problem

At its simplest, the channelization problem features the given data M, N, and A—the number of users, the number of information regions, and the user information request matrix describing which information is wanted by which user. A solution to the problem finds K, C, and S—the number of channels, the channel information matrix describing which information goes onto which channels, and the user subscription matrix describing which users listen to which channels. As stated, there is no unique solution to the problem, many solutions are feasible. A feasible solution here means that each user gets all the information that was requested.

Besides being feasible, there are other considerations for channelization solutions. Some times these considerations may involve hard limits (inequality constraints), while at other times they involve softer goals (some function to be optimized). Possible limits on system or user information density were mentioned above. There could be limits on the number of channels per user or on the total number of channels.

Generally it is considered better if

- the number of total channels is lower
- the number of channels received by any user is lower
- the system information density is lower
- the information density to any user is lower

- the value of information to any user is higher
- the total value of information to all users is higher

The view here of the channelization problem is a static one, where the users and their profiles are unchanging, and, thus, the information regions are unchanging. In the "real-world" problem, the number of users and their requests change with time, and the channelization problem is frequently reworked. It may be practically inconvenient, if not impossible, to totally rework the problem for each new user request, so there is an interest in algorithms that modify a prior solution when a new user request is received or when a user request is withdrawn.

Preliminaries

There are some properties of the user request matrix in a "well-posed" channelization problem. No entire row of the request matrix may be zero, for that would mean a user who has requested no information, and such a user simply could be dropped from consideration. No entire column of the user request matrix is zero, for that would mean an information region requested by no user, and such an information region simply could be dropped from consideration. Of course, no such region would be created by the factoring process.

Another impossibility in a well-posed problem is that two or more columns of the request matrix are identical. This would imply two or more information regions that are "totally correlated"—requested by the same subset of users. Since the information regions are formed by factoring user information profiles, such a set of regions would never be created. Instead, they would be lumped into one, combined region. It is possible that two or more rows of the request matrix are equal. This means two or more users with identical information requests.

It is sometimes illuminating to experiment with the channelization problem in a computer simulation where the number of users and the number of information regions are specified, and the binary user request matrix is filled with 1s and 0s in some random manner. Such experimentation does not necessarily lead to a well-posed problem. A matrix randomly filled with 1s and 0s may well have all-zero rows or all-zero columns or equal columns.

One general approach to the problem is to initialize the channelization matrix equal to the user request matrix—the so-called "one-channel-per-user" singlecast solution mentioned below, and then find algorithms that modify the channelization and subscription matrices to improve the solution. During this process, it is entirely possible, even desirable, that an entire row of the channelization matrix becomes zero, meaning a channel with no information—a channel that disappears. It is never possible in a feasible solution that an entire column of the channelization matrix might vanish, for that would mean an information region available to no channel. Since, as mentioned above, every information region was requested by some user, every information region must be included on some channel.

Simple Solutions

The one-channel-per-region solution. One trivial solution is to create a single channel for each information region and send that channel to each user who requests the region. That is,

$$K = N, \quad C = I_N, \quad S = A \tag{14}$$

where I_N is the $N \times N$ identity matrix. In this solution, the total system information density is minimized, and no user gets any excess information, but the number of channels and their management may well be excessive.

The one-channel-per-user solution. Another simple solution is to create a separate channel for each user and put on it only what that user has requested. This solution is

$$K = M, \quad C = A, \quad S = I_M \tag{15}$$

This one-channel-per-user singlecast solution assures that no user gets any excess information, but the total system information load will likely be very high.

The one-channel solution. Another trivial solution is the one-channel solution. All the information regions are placed onto a single channel, and every user subscribes to that channel. This solution is

$$K = 1, \quad C = 1_N^T, \quad S = 1_M \tag{16}$$

This solution just passes along the problem of sorting out the information to the ultimate users, who may lack the resources to accomplish its solution. The expectation is that there is so much total information that no user has the resources to receive it all, store it, and decide what is valuable. The whole point of the channelizing is to place this burden on some central network authority, removing it from the users.

Manipulating the Channelization and Subscription Matrices

Channelization algorithms make changes to the channelization matrix and to the subscription matrix. Sometimes these changes reduce the number of channels by eliminating some channels. Sometimes the channels must be put into a particular order to implement an algorithm—"smallest" channel first, say. As a practical matter, it is often not a good idea to actually change the allocated size of the channelization matrix or to rearrange the storage order of its rows. Instead, it is usually efficient to establish and maintain one or more "ordering" arrays which serve to indirectly address rows of the channelization matrix in some specified order, or to allow rows to be disregarded entirely.

For example, the number of information regions per channel might be used to order the channels (rows), with a 0 or -1 entry used to indicate "dead" channels that have been removed. Two integer arrays of length K might be established. One, a count array, has the count of the number of information regions contained in the kth channel, and the other, an order array, has a list of numbers from 1 to K, rearranged so that the jth element of the order array is the row number of the jth "smallest" channel by count. This method will usually improve the efficiency of a channelization algorithm.

No-cost improvements

One approach to solving the channelization problem might be to start with some initial solution, perhaps one of those trivial solutions given above, and then improve that solution by revising the channelization and subscription matrices.

It could happen that two or more channels contain the identical information—the same set of information regions. In this case, it is a simple matter to eliminate all but one of the identical channels, and resubscribe all users that got any of them to the remaining one.

Unless the problem involves information utility considerations that are not identical, it seems reasonable to assign all users with identical requests to the same subscription of channels. In the case of "one channel per user", this provides a simplification of not creating multiple identical channels, although such simplification is not easily expressible in matrix symbolic terms. The result would be "one channel per unique user request". If we denote by

$$A' = \text{ndup}(A) \tag{17}$$

the result of the operation of striking out from the matrix A all rows that are equal to any row above them, then it is easy to see that $C = A'$ and K is the number of rows remaining in C, but it is not easy to express the subscription matrix, which will no longer be square, although it will have a single 1 in each row. The way to create the S matrix is to begin with the $M \times M$ identity matrix, and whenever row m of A is deleted because it is equal to some preceding row p, then column m of S should be deleted, while column p should have a 1 placed into the mth row.

containment Algorithms

Direct containment. A more complex algorithm is called the "containment algorithm". Begin with any channelization and subscription matrices. Examine the rows of the channelization matrix to see if any row is "contained" by another. Row j "contains" row k if every element of row j equals or exceeds its counterpart in row k. For the binary channelization matrix, this means that row j has a 1 in each column where row k has a 1. When one row contains another, one channel has every information region that is included in another. In the containment algorithm, when a row of the channelization matrix is contained, it is removed, and users previously subscribing to that channel are subscribed instead to the containing channel. This containment may chain. That is, channel i may be contained by channel j, which, in turn, is contained by channel k, and so on. All these channels are ultimately replaced by the single channel that contains them all.

The resulting information channelization (the number of channels and which channels carry which information regions) is unique, but the user subscription matrix is not necessarily unique. It is quite possible that two rows, neither of which contains the other, each contain a third, and the users of that third row may be reassigned to either of the first two channel rows. The channel selected depends on the details of the algorithm. The result is that each user subscribes to a single channel, but there may be fewer channels than users, and some users receive excess information. To reduce that excess information it is useful to organize the algorithm so that smaller rows (rows having fewer 1s) are considered before larger rows when searching for containments. This assures that the smaller of two rows that contain a third will be chosen to replace the contained row. It also speeds up the algorithm, for a smaller row can never contain a larger row, and these cases never need be examined.

When several channels each carry most of the information regions to begin with, what may happen is that these channels contain all the others, and the result is the one-channel-with-everything case. When there are many more information regions than there are users, the number of containments becomes vanishingly small, and the containment algorithm does nothing.

Reverse containment. The concept of containment may be used for a different kind of algorithm. Begin with any channelization and user subscription matrices. For example, start with the one-channel-per-user approach. Then search for channels that are contained by other channels. When channel i is contained by channel j, remove row i from row j of the channelization matrix C (remove all the 1s in row i from row j), and then go down column i of the subscription matrix, placing 1s as necessary to assure that each user previously subscribing to channel j now subscribes to both channels i and j. That is, once i and j are found, for $n = 1, \ldots, N$, set C_{jn} to 0 whenever C_{in} is 1; then for $m = 1, \ldots, M$, set S_{mi} to 1 whenever S_{mj} is 1. This reverse containment algorithm does not, at first, appear to change the number of channels. It simplifies some channels by breaking them into parts, which are then reassembled in the user subscriptions, causing the total system information load to decrease, while not increasing any user load. In fact, the number of channels may decrease, because all information regions may be removed from them. If one channel happened to be the union of two others, then the two smaller channels could be removed from the larger one, leaving nothing.

This algorithm, too, is not completely defined by the description above. Row k may contain both rows i and j (and others). Which, of these contained rows, is removed from row k is a matter of choice, and here it is not easy to see which choice is best. Once a containing channel has been made smaller by removal of a contained channel, then the new smaller channel is likely to be contained by others, and it may be removed from them.

When this algorithm is tested on randomly constructed problems that have fewer information regions than users, especially if the probability that a given user requests a given information region is either low or high, what often happens is that the channelization is initialized to the one-channel-per-user solution, but winds up at the one-channel-per-information-region result—a result having fewer channels and far lower total system load. When there are many more information regions than there are users, the number of containments becomes vanishingly small, and the reverse containment algorithm does nothing.

Clustering Algorithms

Further reductions in the number of channels may be accomplished by an algorithm (or a group of algorithms) called "clustering". In this concept, when two or more user requests differ by little (perhaps as measured by the information density) each of them is assigned to a channel that contains them all, a channel made by unioning the information regions in the nearby requests. Algorithms of this type produce results that depend on the precise ordering of comparisons and replacements. Clustering algorithms maintain the one-channel-per-user result.

EXPERIMENTAL SOFTWARE DESCRIPTION

Introduction

As a part of the investigation of the channelization problem, an experimental test-bed computer program was written to simulate, exercise, and evaluate some candidate channelization algorithms using probabalistic assumptions about user requests for information. This program, named CPcrib, is not a part of the main deliverable software of the project and is not a polished product. It is a flexible and modifiable tool primarily created to be used only by its maker, but possibly of further use. It is described here mostly to enable better understanding and interpretation of its results—this is not a user's manual. In addition to this document, the program code contains much useful comment information, more than might be expected.

The program is a console application that takes user input only from the command line and writes its major outputs to a log file named CPcrib.log, although some error indications and minor output are written to the screen. The user generally specifies a probabalistic mode to simulate user information requests, specifies the probability parameters involved, specifies the size of the problem, by giving the number of users and the number of information regions, and specifies a channelization algorithm. Options are available to examine a hard-wired range of probability parameters and/or a hard-wired range of problem sizes in one run. A debug option is also available, although that is primarily useful only during program modification.

When considering the containment algorithm (which was implemented first), the program deals with the "user request matrix", which is a binary matrix where a 1 in row m and column n indicates that user m is requesting information region n. For the other channelization algorithms, the program uses the "channelization matrix", another binary matrix where a "1" in row m and column n indicates that channel m contains information product n. One or the other of these matrices is initialized in some probabalistic way, and a channelization algorithm is exercised that modifies the matrix. Measures of performance are made, before and after channelization, and such measures are averaged to produce results.

the program has several probabalistic modes of operation: a deterministic mode used only with the containment algorithm, and three Monte-Carlo modes used with all algorithms. If the total number of elements in the user request matrix (the number of users times the number of information regions) is quite small, say fewer than 28 or so, then there are only two raised to the power of this number of possibilities for the user request matrix. With 28 elements, there are just over 268 million different possible such matrices. In the deterministic mode, each of these possibilities is examined in turn. The probability of a particular user request matrix occurring is computable by counting the total number of 1s in the matrix, and applying channelization algorithms to each matrix allows result statistics to be computed.

In the deterministic mode, called the "det" mode in the program, each element of the binary matrix is assumed to be an i.i.d. (independent, identically distributed) random variable. The only parameter needed to describe this situation is the probability that any element is "1". The deterministic mode allows a range of several different such probabilities to be examined at once. The same deterministic sequence of matrices is involved, regardless of the probability. Only the weighting factors involved in averaging the performance results change as the probability changes, so several different weighting factors may be considered at once.

Prime interest in the algorithms is on cases with many more than 28 elements in the user request or channelization matrix, in fact, with thousands of elements in the matrix, where the deterministic mode is out of the question. The Monte-Carlo modes fill the matrix according to a specified probability that any element is 1. A channelization algorithm is then applied to the randomly initialized matrix, the performance of the algorithm is assessed, and that assessment is averaged over a specified number of random samples for the initial matrix. The Monte-Carlo modes can only consider a single probability scheme at one time, since the elements in the sample matrices depend on the scheme. The three schemes in the program are discussed below.

Random Binary Matrices

In order to test channelization algorithms it is useful to be able to fill binary matrices, such as the user request matrix A or the channelization matrix C, with random sequences of 1s and 0s. The program does this in one of three different ways.

It is easy to create a stream of binary bits—1s and 0s—that appear independently randomly chosen with a probability of 1 of exactly one-half. Such a stream might represent a sequence of fair coin flips, where 1 means a head, and 0 means a tail. To do this requires an integer random number generator.

The first such integer generator that was used—now supersceded—implemented the mathematical relationship

$$\text{seed} \leftarrow (65539 * \text{seed}) \bmod 2147483648 \qquad (1)$$

via the code

```
seed = 65539 * seed;

if ( seed < 0 )   seed = ( seed + 2147483647 ) + 1;
```

beginning with seed as any odd, positive integer. The multiplier here is two to the sixteenth plus three, while the modulus is two to the thirty-first power. This code is believed to have originated in the dark ages of computing at IBM as a part of the RANDU FORTRAN random number generator, and it depends on thirty-two bit, two's-complement integers and on integer overflow going unnoticed. It is very efficient. Problems were finally noticed with this generator, and a quick survey of the random number generator literature showed that this generator is notoriously bad, so it was replaced.

The replacement code implements the very similar looking mathematical relationship

$$\text{seed} \leftarrow (16807 * \text{seed}) \bmod 2147483647 \qquad (2)$$

The multiplier here is seven to the fifth power, while the modulus is one less than two to the thirty-first power, which is a prime number. The implementation avoids any possibility of integer overflow, as

```
    seed = aa * ( seed % qq ) - rr * ( seed / qq );

    if ( seed < 0 )  seed += mm;
```

where aa is the multiplier 16807; mm is the modulus 2147483647; qq is 127773, which is the integer part of mm divided by aa; and rr is 2836, which is the remainder on dividing mm by aa. This generator is slower than the original one, but it generates much more random low-order bits of the seed, and has a longer period.

Such random integers may be split up into bits to produce a stream of random bits, each of which has a probability of 1 of one-half, and each of which is independent of the others. Of course, the "zeroth" (most significant, top) bit of seed cannot be used. Since seed is always positive, this bit is always 0. The low-order 31 bits are used.

The very first way the program worked (now abandoned) was to use such a stream of bits to fill the binary matrices. This method was very efficient, but very limiting, for the only value of the probability of 1 available was one-half.

The second mode (called the "hex" mode and still available, but outdated) creates from the random bits above a second stream whose probability of 1 is an integral multiple of one-sixteenth. It does this by taking the original bits four-at-a-time, and comparing the resulting random integer (from zero to fifteen) with a fixed integer n_p (from one to fifteen). If the random integer is less than the fixed integer, then the resulting bit is set to 1 (otherwise it is set to 0). The probability that the resulting bit is 1 is n_p divided by sixteen. This method is conceptually simple, easy to implement, fairly efficient, but still limiting. Each random integer generates seven such random bits.

The third mode (called the "gen" mode) allows any probability of 1 to be used, approximating that probability by a thirty bit binary fraction. The problem is to efficiently turn a sequence of equally likely 1s and 0s into a sequence of 1s and 0s where the probability of 1 is the arbitrary p. Let the equally likely binary "input" sequence be denoted as $\{b_1, b_2, \ldots\}$. Let the K bit binary approximate expansion of p be $p = 0.p_1p_2\ldots p_K$. Let the binary "output" sequence be $\{c_1, c_2, \ldots\}$. To generate c_1, simply test the sequence of bs against the sequence of ps, and set c_1 to the first p_k that is not equal to b_k. To generate subsequent cs, repeat the process, renumbering the bs so that the first unused b is b_1.

As an example, suppose eight-bit approximations for probability values are used $(K = 8)$, and $p = \frac{181}{256} = 0.10110101)_b$. Suppose the equally-likely input bit stream is $\{b_k\} = \{10011101001100110100\ldots\}$. First, $p_1 = b_1 = 1$, so b_1 is ignored. Next, $p_2 = b_2 = 0$, so b_2 is also ignored. Next, $p_3 \neq b_3$, so $c_1 = p_3 = 1$. (In case this process goes all the way to $p_K = b_K$, just pick c_1 arbitrarily.) To go on to find c_2, just start over, except discard the first three "used" bits from the b sequence, renumbering the remainder. On average, two (or slightly fewer) bits of the b input sequence will generate one bit of the c

output sequence, whatever number of bits are used to approximate p.

This method is not so conceptually obvious, and is harder to implement than the previous method, but is even more efficient, and may be used for any probability.

The fourth mode (called the "zipf" mode) needs some introduction. The modes above have the same probability of "1" for each element of the matrix, which means that any information region is, on average, as likely to be requested as any other. Real-world requests for information regions, however, are a type of popularity contest. It is typical of such contests that the popularity of a few objects is far higher than that of most others. Statistical studies often show that if the objects of choice are ranked in order of decreasing popularity, then their relative likelihood is roughly proportional to the inverse of the rank order raised to some constant power, a power usually near 1.0. That is, if the popularity rank of an object is r, where $r = 1, 2, \ldots$, then the probability of selecting that object is

$$p_r = \frac{c}{r^\beta} \tag{3}$$

where the constant c is between zero and one, and the exponent β is near one. This probability law is often called Zipf's law, after a Harvard linguist who used it to model the occurrence of words in English text. It has also been used to model the distribution of city populations (people choosing homes), of corporate incomes (dollars choosing homes), and of website popularity (users choosing information regions).

Taking logarithms

$$\log p_r = \log c - \beta \log r \tag{4}$$

so that Zipf's law graphs as a straight line on log-log paper. Of course, almost every reasonable relationship graphs as a straight line on log-log paper.

Assuming that the information regions are ranked according to their popularity, Zipf's law may be used to randomly fill the user request or the initial channelization matrix. It would be convenient if the summation

$$s_n(\beta) = \sum_{r=1}^{n} \frac{1}{r^\beta} \tag{5}$$

could be expressed in some simple closed-form way, but it cannot. This sum is proportional to the expected number of the objects of rank 1 through n. For exponent β less than or equal to one, the sum diverges with increasing n.